

libstdc++

Generated by Doxygen 1.16.1

1 Deprecated List	1
2 Todo List	2
3 Topic Documentation	4
3.1 Algorithms	4
3.1.1 Detailed Description	4
3.1.2 Generalized Numeric operations	5
3.1.3 Mutating	21
3.1.4 Non-Mutating	43
3.1.5 Sorting	62
3.2 Atomics	105
3.2.1 Detailed Description	109
3.2.2 Macro Definition Documentation	109
3.2.3 Typedef Documentation	110
3.2.4 Enumeration Type Documentation	116
3.2.5 Function Documentation	116
3.3 Concurrency	117
3.3.1 Detailed Description	117
3.3.2 Condition Variables	117
3.3.3 Futures	118
3.3.4 Mutexes	122
3.3.5 Threads	125
3.4 Containers	126
3.4.1 Detailed Description	126
3.4.2 Associative	127
3.4.3 Sequences	127
3.4.4 Unordered Associative	128
3.5 Diagnostics	129
3.5.1 Detailed Description	130
3.5.2 Function Documentation	130
3.5.3 Exceptions	133
3.6 Extensions	138
3.6.1 Detailed Description	139
3.6.2 Dynamic Bitset	139
3.6.3 Policy-Based Data Structures	144
3.6.4 SGI	153
3.7 Filesystem	162
3.7.1 Detailed Description	167
3.7.2 Typedef Documentation	167

3.7.3 Enumeration Type Documentation	167
3.7.4 Function Documentation	167
3.8 I/O	168
3.8.1 Detailed Description	170
3.8.2 Typedef Documentation	170
3.9 Iterators	172
3.9.1 Detailed Description	176
3.9.2 Function Documentation	176
3.9.3 Iterator Tags	178
3.10 Locales	178
3.10.1 Detailed Description	179
3.10.2 Function Documentation	179
3.11 Numerics	181
3.11.1 Detailed Description	181
3.11.2 Bit manipulation	182
3.11.3 Complex Numbers	182
3.11.4 Decimal Floating-Point Arithmetic	191
3.11.5 Mathematical Special Functions	192
3.11.6 Numeric Arrays	221
3.11.7 Random Number Generation	237
3.11.8 TR1 Mathematical Special Functions	251
3.12 Ranges	256
3.12.1 Detailed Description	256
3.13 Regular Expressions	257
3.13.1 Detailed Description	260
3.13.2 Typedef Documentation	260
3.13.3 Function Documentation	261
3.13.4 Base and Implementation Classes	278
3.14 Strings	279
3.14.1 Detailed Description	279
3.14.2 Typedef Documentation	279
3.15 Technical Specifications	280
3.15.1 Detailed Description	280
3.15.2 Filesystem TS	281
3.15.3 Library Fundamentals TS	286
3.15.4 Parallelism TS	301
3.16 Utilities	302
3.16.1 Detailed Description	305
3.16.2 Function Documentation	305

3.16.3 Variable Documentation	314
3.16.4 Concepts	315
3.16.5 Function Objects	316
3.16.6 Memory	326
3.16.7 Metaprogramming	355
3.16.8 Rational Arithmetic	364
3.16.9 Time	366
4 Namespace Documentation	387
4.1 <code>__gnu_cxx</code> Namespace Reference	387
4.1.1 Detailed Description	404
4.1.2 Typedef Documentation	404
4.1.3 Function Documentation	404
4.2 <code>__gnu_cxx::__detail</code> Namespace Reference	414
4.2.1 Detailed Description	414
4.2.2 Function Documentation	415
4.3 <code>__gnu_cxx::typelist</code> Namespace Reference	415
4.3.1 Detailed Description	415
4.3.2 Function Documentation	416
4.4 <code>__gnu_debug</code> Namespace Reference	416
4.4.1 Detailed Description	421
4.4.2 Typedef Documentation	422
4.4.3 Enumeration Type Documentation	422
4.4.4 Function Documentation	422
4.5 <code>__gnu_internal</code> Namespace Reference	424
4.5.1 Detailed Description	424
4.6 <code>__gnu_parallel</code> Namespace Reference	424
4.6.1 Detailed Description	431
4.6.2 Typedef Documentation	432
4.6.3 Enumeration Type Documentation	432
4.6.4 Function Documentation	433
4.6.5 Variable Documentation	473
4.7 <code>__gnu_pbds</code> Namespace Reference	474
4.7.1 Detailed Description	476
4.8 <code>__gnu_sequential</code> Namespace Reference	476
4.8.1 Detailed Description	476
4.9 <code>abi</code> Namespace Reference	476
4.9.1 Detailed Description	476
4.10 <code>std</code> Namespace Reference	476

4.10.1 Detailed Description	590
4.10.2 Typedef Documentation	590
4.10.3 Enumeration Type Documentation	592
4.10.4 Function Documentation	593
4.10.5 Variable Documentation	657
4.11 std::__debug Namespace Reference	658
4.11.1 Detailed Description	665
4.11.2 Function Documentation	665
4.12 std::__detail Namespace Reference	665
4.12.1 Detailed Description	668
4.12.2 Function Documentation	668
4.13 std::__parallel Namespace Reference	669
4.13.1 Detailed Description	686
4.13.2 Function Documentation	686
4.14 std::chrono Namespace Reference	686
4.14.1 Detailed Description	694
4.15 std::decimal Namespace Reference	694
4.15.1 Detailed Description	703
4.15.2 Function Documentation	703
4.16 std::experimental Namespace Reference	703
4.16.1 Detailed Description	713
4.16.2 Function Documentation	714
4.16.3 Variable Documentation	715
4.17 std::filesystem Namespace Reference	715
4.17.1 Detailed Description	719
4.18 std::literals Namespace Reference	719
4.18.1 Detailed Description	720
4.19 std::literals::chrono_literals Namespace Reference	720
4.19.1 Detailed Description	720
4.20 std::placeholders Namespace Reference	721
4.20.1 Detailed Description	721
4.21 std::regex_constants Namespace Reference	721
4.21.1 Detailed Description	723
4.21.2 Enumeration Type Documentation	723
4.21.3 Function Documentation	723
4.21.4 Variable Documentation	727
4.22 std::rel_ops Namespace Reference	731
4.22.1 Detailed Description	731
4.22.2 Function Documentation	731

4.23 <code>std::this_thread</code> Namespace Reference	733
4.23.1 Detailed Description	733
4.23.2 Function Documentation	733
4.24 <code>std::tr1</code> Namespace Reference	734
4.24.1 Detailed Description	737
4.24.2 Function Documentation	737
4.25 <code>std::tr1::__detail</code> Namespace Reference	739
4.25.1 Detailed Description	739
4.26 <code>std::tr2</code> Namespace Reference	739
4.26.1 Detailed Description	741
4.27 <code>std::tr2::__detail</code> Namespace Reference	741
4.27.1 Detailed Description	741
5 Concept Documentation	741
5.1 <code>std::assignable_from</code> Concept Reference	741
5.1.1 Concept definition	741
5.1.2 Detailed Description	741
5.2 <code>std::common_reference_with</code> Concept Reference	741
5.2.1 Concept definition	742
5.2.2 Detailed Description	742
5.3 <code>std::common_with</code> Concept Reference	742
5.3.1 Concept definition	742
5.3.2 Detailed Description	742
5.4 <code>std::constructible_from</code> Concept Reference	742
5.4.1 Concept definition	742
5.4.2 Detailed Description	742
5.5 <code>std::convertible_to</code> Concept Reference	742
5.5.1 Concept definition	742
5.5.2 Detailed Description	742
5.6 <code>std::copy_constructible</code> Concept Reference	743
5.6.1 Concept definition	743
5.6.2 Detailed Description	743
5.7 <code>std::default_initializable</code> Concept Reference	743
5.7.1 Concept definition	743
5.7.2 Detailed Description	743
5.8 <code>std::derived_from</code> Concept Reference	743
5.8.1 Concept definition	743
5.8.2 Detailed Description	743
5.9 <code>std::destructible</code> Concept Reference	743

5.9.1 Concept definition	743
5.9.2 Detailed Description	743
5.10 <code>std::equivalence_relation</code> Concept Reference	744
5.10.1 Concept definition	744
5.10.2 Detailed Description	744
5.11 <code>std::indirectly_comparable</code> Concept Reference	744
5.11.1 Concept definition	744
5.11.2 Detailed Description	744
5.12 <code>std::indirectly_copyable</code> Concept Reference	744
5.12.1 Concept definition	744
5.12.2 Detailed Description	744
5.13 <code>std::indirectly_movable</code> Concept Reference	744
5.13.1 Concept definition	744
5.13.2 Detailed Description	744
5.14 <code>std::indirectly_readable</code> Concept Reference	744
5.14.1 Concept definition	745
5.14.2 Detailed Description	745
5.15 <code>std::indirectly_swappable</code> Concept Reference	745
5.15.1 Concept definition	745
5.15.2 Detailed Description	745
5.16 <code>std::indirectly_writable</code> Concept Reference	745
5.16.1 Concept definition	745
5.16.2 Detailed Description	745
5.17 <code>std::invocable</code> Concept Reference	745
5.17.1 Concept definition	745
5.17.2 Detailed Description	745
5.18 <code>std::mergeable</code> Concept Reference	746
5.18.1 Concept definition	746
5.18.2 Detailed Description	746
5.19 <code>std::move_constructible</code> Concept Reference	746
5.19.1 Concept definition	746
5.19.2 Detailed Description	746
5.20 <code>std::permutable</code> Concept Reference	746
5.20.1 Concept definition	746
5.20.2 Detailed Description	746
5.21 <code>std::predicate</code> Concept Reference	746
5.21.1 Concept definition	746
5.21.2 Detailed Description	746
5.22 <code>std::ranges::bidirectional_range</code> Concept Reference	747

5.22.1 Concept definition	747
5.22.2 Detailed Description	747
5.23 <code>std::ranges::borrowed_range</code> Concept Reference	747
5.23.1 Concept definition	747
5.23.2 Detailed Description	747
5.24 <code>std::ranges::common_range</code> Concept Reference	747
5.24.1 Concept definition	747
5.24.2 Detailed Description	747
5.25 <code>std::ranges::contiguous_range</code> Concept Reference	747
5.25.1 Concept definition	747
5.25.2 Detailed Description	747
5.26 <code>std::ranges::forward_range</code> Concept Reference	747
5.26.1 Concept definition	748
5.26.2 Detailed Description	748
5.27 <code>std::ranges::input_range</code> Concept Reference	748
5.27.1 Concept definition	748
5.27.2 Detailed Description	748
5.28 <code>std::ranges::output_range</code> Concept Reference	748
5.28.1 Concept definition	748
5.28.2 Detailed Description	748
5.29 <code>std::ranges::random_access_range</code> Concept Reference	748
5.29.1 Concept definition	748
5.29.2 Detailed Description	748
5.30 <code>std::ranges::range</code> Concept Reference	748
5.30.1 Concept definition	749
5.30.2 Detailed Description	749
5.31 <code>std::ranges::sized_range</code> Concept Reference	749
5.31.1 Concept definition	749
5.31.2 Detailed Description	749
5.32 <code>std::ranges::view</code> Concept Reference	749
5.32.1 Concept definition	749
5.32.2 Detailed Description	749
5.33 <code>std::ranges::viewable_range</code> Concept Reference	749
5.33.1 Concept definition	749
5.33.2 Detailed Description	749
5.34 <code>std::regular_invocable</code> Concept Reference	749
5.34.1 Concept definition	750
5.34.2 Detailed Description	750
5.35 <code>std::relation</code> Concept Reference	750

5.35.1 Concept definition	750
5.35.2 Detailed Description	750
5.36 <code>std::same_as</code> Concept Reference	750
5.36.1 Concept definition	750
5.36.2 Detailed Description	750
5.37 <code>std::simd::__vec_builtin</code> Concept Reference	750
5.37.1 Concept definition	750
5.37.2 Detailed Description	750
5.38 <code>std::simd::__vec_builtin_of</code> Concept Reference	750
5.38.1 Concept definition	751
5.38.2 Detailed Description	751
5.39 <code>std::sortable</code> Concept Reference	751
5.39.1 Concept definition	751
5.39.2 Detailed Description	751
5.40 <code>std::strict_weak_order</code> Concept Reference	751
5.40.1 Concept definition	751
5.40.2 Detailed Description	751
5.41 <code>std::uniform_random_bit_generator</code> Concept Reference	751
5.41.1 Concept definition	751
5.41.2 Detailed Description	751
5.42 <code>std::weakly_incrementable</code> Concept Reference	752
5.42.1 Concept definition	752
5.42.2 Detailed Description	752
6 Class Documentation	752
6.1 <code>__gnu_parallel::__accumulate_binop_reduct<_BinOp></code> Struct Template Reference	752
6.1.1 Detailed Description	752
6.2 <code>__gnu_parallel::__accumulate_selector<_It></code> Struct Template Reference	752
6.2.1 Detailed Description	753
6.2.2 Member Function Documentation	753
6.2.3 Member Data Documentation	754
6.3 <code>__gnu_parallel::__adjacent_difference_selector<_It></code> Struct Template Reference	754
6.3.1 Detailed Description	755
6.3.2 Member Data Documentation	755
6.4 <code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	755
6.4.1 Detailed Description	755
6.4.2 Member Function Documentation	756
6.5 <code>__gnu_cxx::__alloc_traits<_Alloc, typename></code> Struct Template Reference	756
6.5.1 Detailed Description	758

6.5.2 Member Typedef Documentation	758
6.5.3 Member Function Documentation	759
6.6 std::__basic_future< _Res > Class Template Reference	763
6.6.1 Detailed Description	764
6.6.2 Member Function Documentation	765
6.7 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference	765
6.7.1 Detailed Description	766
6.7.2 Member Typedef Documentation	766
6.8 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference	766
6.8.1 Detailed Description	767
6.8.2 Member Typedef Documentation	767
6.9 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference	767
6.9.1 Detailed Description	768
6.9.2 Member Function Documentation	768
6.10 __gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference	771
6.10.1 Detailed Description	771
6.11 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference	771
6.11.1 Detailed Description	772
6.11.2 Member Function Documentation	772
6.11.3 Member Data Documentation	773
6.12 __gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference	773
6.12.1 Detailed Description	774
6.12.2 Member Function Documentation	774
6.12.3 Member Data Documentation	774
6.13 std::__ctype_abstract_base< _CharT > Class Template Reference	774
6.13.1 Detailed Description	776
6.13.2 Member Typedef Documentation	776
6.13.3 Member Function Documentation	777
6.14 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference	788
6.14.1 Detailed Description	790
6.14.2 Member Data Documentation	790
6.15 __gnu_parallel::__fill_selector< _It > Struct Template Reference	790
6.15.1 Detailed Description	790
6.15.2 Member Function Documentation	791
6.15.3 Member Data Documentation	791
6.16 __gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference	791
6.16.1 Detailed Description	792
6.16.2 Member Function Documentation	792

6.17	__gnu_parallel::__find_if_selector Struct Reference	793
6.17.1	Detailed Description	793
6.17.2	Member Function Documentation	793
6.18	__gnu_parallel::__for_each_selector< _It > Struct Template Reference	794
6.18.1	Detailed Description	795
6.18.2	Member Function Documentation	795
6.18.3	Member Data Documentation	796
6.19	__cxxabiv1::__forced_unwind Class Reference	796
6.19.1	Detailed Description	796
6.20	__gnu_parallel::__generate_selector< _It > Struct Template Reference	796
6.20.1	Detailed Description	797
6.20.2	Member Function Documentation	797
6.20.3	Member Data Documentation	797
6.21	__gnu_parallel::__generic_find_selector Struct Reference	797
6.21.1	Detailed Description	798
6.22	__gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference	798
6.22.1	Detailed Description	800
6.22.2	Member Data Documentation	800
6.23	__gnu_parallel::__identity_selector< _It > Struct Template Reference	800
6.23.1	Detailed Description	800
6.23.2	Member Function Documentation	801
6.23.3	Member Data Documentation	801
6.24	__gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference	801
6.24.1	Detailed Description	802
6.24.2	Constructor & Destructor Documentation	802
6.24.3	Member Function Documentation	803
6.24.4	Member Data Documentation	803
6.25	std::__is_fast_hash< _Hash > Struct Template Reference	803
6.25.1	Detailed Description	804
6.26	std::__is_location_invariant< _Tp > Struct Template Reference	804
6.26.1	Detailed Description	804
6.27	__gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	804
6.27.1	Detailed Description	804
6.28	__gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	804
6.28.1	Detailed Description	805
6.29	__gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference	805
6.29.1	Detailed Description	805
6.30	__gnu_parallel::__mismatch_selector Struct Reference	806
6.30.1	Detailed Description	806

6.30.2 Member Function Documentation	806
6.31 <code>__gnu_cxx::__mt_alloc< _Tp, _Poolp ></code> Class Template Reference	807
6.31.1 Detailed Description	808
6.32 <code>__gnu_cxx::__mt_alloc_base< _Tp ></code> Class Template Reference	808
6.32.1 Detailed Description	809
6.33 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	809
6.33.1 Detailed Description	810
6.34 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	810
6.34.1 Detailed Description	810
6.35 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	810
6.35.1 Detailed Description	810
6.36 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	811
6.36.1 Detailed Description	811
6.37 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	811
6.37.1 Detailed Description	811
6.38 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	811
6.38.1 Detailed Description	812
6.39 <code>std::__new_allocator< _Tp ></code> Class Template Reference	812
6.39.1 Detailed Description	813
6.40 <code>std::__numeric_limits_base</code> Struct Reference	813
6.40.1 Detailed Description	815
6.40.2 Member Data Documentation	815
6.41 <code>__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread ></code> Struct Template Reference	818
6.41.1 Detailed Description	818
6.42 <code>__gnu_cxx::__pool< _Thread ></code> Class Template Reference	818
6.42.1 Detailed Description	818
6.43 <code>__gnu_cxx::__pool< false ></code> Class Reference	818
6.43.1 Detailed Description	819
6.44 <code>__gnu_cxx::__pool< true ></code> Class Reference	819
6.44.1 Detailed Description	820
6.45 <code>__gnu_cxx::__pool_alloc< _Tp ></code> Class Template Reference	820
6.45.1 Detailed Description	821
6.46 <code>__gnu_cxx::__pool_alloc_base</code> Class Reference	821
6.46.1 Detailed Description	822
6.47 <code>__gnu_cxx::__pool_base</code> Struct Reference	822

6.47.1 Detailed Description	823
6.48 <code>__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc></code> Class Template Reference	823
6.48.1 Detailed Description	826
6.49 <code>std::tr2::__reflection_typelist<_Elements></code> Struct Template Reference	826
6.49.1 Detailed Description	826
6.50 <code>std::tr2::__reflection_typelist<_First, _Rest...></code> Struct Template Reference	826
6.50.1 Detailed Description	826
6.51 <code>std::tr2::__reflection_typelist<></code> Struct Reference	827
6.51.1 Detailed Description	827
6.52 <code>__gnu_parallel::__replace_if_selector<_It, _Op, _Tp></code> Struct Template Reference	827
6.52.1 Detailed Description	827
6.52.2 Constructor & Destructor Documentation	828
6.52.3 Member Function Documentation	828
6.52.4 Member Data Documentation	828
6.53 <code>__gnu_parallel::__replace_selector<_It, _Tp></code> Struct Template Reference	829
6.53.1 Detailed Description	829
6.53.2 Constructor & Destructor Documentation	829
6.53.3 Member Function Documentation	830
6.53.4 Member Data Documentation	830
6.54 <code>__gnu_cxx::__scoped_lock</code> Class Reference	830
6.54.1 Detailed Description	830
6.55 <code>__gnu_parallel::__transform1_selector<_It></code> Struct Template Reference	831
6.55.1 Detailed Description	831
6.55.2 Member Function Documentation	831
6.55.3 Member Data Documentation	832
6.56 <code>__gnu_parallel::__transform2_selector<_It></code> Struct Template Reference	832
6.56.1 Detailed Description	833
6.56.2 Member Function Documentation	833
6.56.3 Member Data Documentation	833
6.57 <code>__gnu_parallel::__unary_negate<_Predicate, argument_type></code> Class Template Reference	833
6.57.1 Detailed Description	834
6.57.2 Member Typedef Documentation	834
6.58 <code>__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base></code> Class Template Reference	835
6.58.1 Detailed Description	838
6.58.2 Constructor & Destructor Documentation	838
6.58.3 Member Function Documentation	841
6.58.4 Member Data Documentation	887
6.59 <code>__gnu_debug::__After_nth_from<_Iterator></code> Class Template Reference	888
6.59.1 Detailed Description	888

6.60	std::_Base_bitset< _Nw > Struct Template Reference	888
6.60.1	Detailed Description	889
6.60.2	Member Data Documentation	889
6.61	std::_Base_bitset< 0 > Struct Reference	889
6.61.1	Detailed Description	890
6.61.2	Member Data Documentation	890
6.62	std::_Base_bitset< 1 > Struct Reference	890
6.62.1	Detailed Description	891
6.63	__gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference	891
6.63.1	Detailed Description	892
6.64	std::_Bind< _Signature > Class Template Reference	892
6.64.1	Detailed Description	892
6.65	std::_Bind_result< _Result, _Signature > Class Template Reference	892
6.65.1	Detailed Description	892
6.66	__gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference	892
6.66.1	Detailed Description	892
6.67	std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference	893
6.67.1	Detailed Description	893
6.68	__gnu_cxx::_Caster< _ToType > Struct Template Reference	893
6.68.1	Detailed Description	893
6.69	__gnu_cxx::_Char_types< _CharT > Struct Template Reference	894
6.69.1	Detailed Description	894
6.70	__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference	894
6.70.1	Detailed Description	896
6.71	std::__detail::_Compiler< _TraitsT > Class Template Reference	896
6.71.1	Detailed Description	897
6.72	std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference	897
6.72.1	Detailed Description	897
6.73	std::_Deque_base< _Tp, _Alloc > Class Template Reference	897
6.73.1	Detailed Description	898
6.73.2	Member Function Documentation	898
6.74	std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	899
6.74.1	Detailed Description	900
6.74.2	Member Function Documentation	900
6.75	__gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference	900
6.75.1	Detailed Description	901
6.75.2	Constructor & Destructor Documentation	901
6.75.3	Member Data Documentation	901

6.76	__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator > Struct Template Reference	902
6.76.1	Detailed Description	902
6.76.2	Member Data Documentation	902
6.77	__gnu_parallel::_DummyReduct Struct Reference	903
6.77.1	Detailed Description	903
6.78	__gnu_debug::_Equal_to<_Type > Class Template Reference	903
6.78.1	Detailed Description	904
6.79	__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare > Class Template Reference	904
6.79.1	Detailed Description	904
6.79.2	Member Typedef Documentation	905
6.80	__gnu_parallel::_EqualTo<_T1, _T2 > Struct Template Reference	905
6.80.1	Detailed Description	906
6.80.2	Member Typedef Documentation	906
6.81	std::__detail::_Executer<_Biliter, _Alloc, _TraitsT > Class Template Reference	906
6.81.1	Detailed Description	907
6.82	__gnu_cxx::_ExtPtr_allocator<_Tp > Class Template Reference	907
6.82.1	Detailed Description	908
6.83	__gnu_cxx::__detail::_Ffit_finder<_Tp > Class Template Reference	908
6.83.1	Detailed Description	908
6.84	std::_Function_base Class Reference	908
6.84.1	Detailed Description	909
6.85	std::_Fwd_list_base<_Tp, _Alloc > Struct Template Reference	909
6.85.1	Detailed Description	911
6.86	std::_Fwd_list_const_iterator<_Tp > Struct Template Reference	911
6.86.1	Detailed Description	911
6.86.2	Friends And Related Symbol Documentation	912
6.87	std::_Fwd_list_iterator<_Tp > Struct Template Reference	912
6.87.1	Detailed Description	912
6.87.2	Friends And Related Symbol Documentation	913
6.88	std::_Fwd_list_node<_Tp > Struct Template Reference	913
6.88.1	Detailed Description	914
6.89	std::_Fwd_list_node_base Struct Reference	914
6.89.1	Detailed Description	915
6.90	__gnu_parallel::_GuardedIterator<_RAIter, _Compare > Class Template Reference	915
6.90.1	Detailed Description	915
6.90.2	Constructor & Destructor Documentation	915
6.90.3	Member Function Documentation	916
6.90.4	Friends And Related Symbol Documentation	916
6.91	__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata > Struct Template Reference	917

6.91.1 Detailed Description	918
6.92 __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata> Struct Template Reference	918
6.92.1 Detailed Description	919
6.93 __gnu_cxx::_Invalid_type Struct Reference	920
6.93.1 Detailed Description	920
6.94 __gnu_pbds::detail::pat_trie_base::_Iter<Node, Leaf, Head, Inode, Is_Forward_Iterator> Class Template Reference	920
6.94.1 Detailed Description	921
6.95 std::__fwlist::_Iterator<_Const, _Ptr> Class Template Reference	921
6.95.1 Detailed Description	922
6.95.2 Friends And Related Symbol Documentation	922
6.96 __gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory> Class Template Reference	922
6.96.1 Detailed Description	925
6.96.2 Member Typedef Documentation	925
6.96.3 Member Function Documentation	925
6.96.4 Friends And Related Symbol Documentation	927
6.96.5 Member Data Documentation	928
6.97 __gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory> Class Template Reference	929
6.97.1 Detailed Description	929
6.98 __gnu_parallel::_Job<_DifferenceTp> Struct Template Reference	929
6.98.1 Detailed Description	930
6.98.2 Member Data Documentation	930
6.99 __gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata> Struct Template Reference	930
6.99.1 Detailed Description	931
6.100 __gnu_parallel::_Less<_T1, _T2> Struct Template Reference	932
6.100.1 Detailed Description	932
6.100.2 Member Typedef Documentation	932
6.101 __gnu_parallel::_Lexicographic<_T1, _T2, _Compare> Class Template Reference	933
6.101.1 Detailed Description	933
6.101.2 Member Typedef Documentation	933
6.102 __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare> Class Template Reference	934
6.102.1 Detailed Description	935
6.102.2 Member Typedef Documentation	935
6.103 std::_List_base<_Tp, _Alloc> Class Template Reference	935
6.103.1 Detailed Description	936
6.104 std::_List_const_iterator<_Tp> Struct Template Reference	937
6.104.1 Detailed Description	937
6.105 std::_List_iterator<_Tp> Struct Template Reference	937
6.105.1 Detailed Description	938

6.106	std::__List_node< _Tp > Struct Template Reference	938
6.106.1	Detailed Description	939
6.107	std::__detail::__List_node_base Struct Reference	940
6.107.1	Detailed Description	940
6.108	std::__detail::__List_node_header Struct Reference	941
6.108.1	Detailed Description	941
6.109	__gnu_parallel::__LoserTreeBase< _Tp, _Compare >::__Loser Struct Reference	942
6.109.1	Detailed Description	942
6.109.2	Member Data Documentation	942
6.110	__gnu_parallel::__LoserTreePointerBase< _Tp, _Compare >::__Loser Struct Reference	942
6.110.1	Detailed Description	942
6.111	__gnu_parallel::__LoserTree< __stable, _Tp, _Compare > Class Template Reference	943
6.111.1	Detailed Description	943
6.111.2	Member Function Documentation	943
6.111.3	Member Data Documentation	944
6.112	__gnu_parallel::__LoserTree< false, _Tp, _Compare > Class Template Reference	944
6.112.1	Detailed Description	945
6.112.2	Member Function Documentation	945
6.112.3	Member Data Documentation	946
6.113	__gnu_parallel::__LoserTreeBase< _Tp, _Compare > Class Template Reference	947
6.113.1	Detailed Description	947
6.113.2	Constructor & Destructor Documentation	948
6.113.3	Member Function Documentation	948
6.113.4	Member Data Documentation	949
6.114	__gnu_parallel::__LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference	949
6.114.1	Detailed Description	950
6.115	__gnu_parallel::__LoserTreePointer< false, _Tp, _Compare > Class Template Reference	950
6.115.1	Detailed Description	951
6.116	__gnu_parallel::__LoserTreePointerBase< _Tp, _Compare > Class Template Reference	951
6.116.1	Detailed Description	952
6.117	__gnu_parallel::__LoserTreePointerUnguarded< __stable, _Tp, _Compare > Class Template Reference	952
6.117.1	Detailed Description	953
6.118	__gnu_parallel::__LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference	953
6.118.1	Detailed Description	954
6.119	__gnu_parallel::__LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference	954
6.119.1	Detailed Description	955
6.120	__gnu_parallel::__LoserTreeTraits< _Tp > Struct Template Reference	955
6.120.1	Detailed Description	956
6.120.2	Member Data Documentation	956

6.121	__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference	956
6.121.1	Detailed Description	957
6.122	__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference	957
6.122.1	Detailed Description	958
6.123	__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference	958
6.123.1	Detailed Description	959
6.124	__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference	959
6.124.1	Detailed Description	959
6.125	__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference	959
6.125.1	Detailed Description	960
6.126	__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference	960
6.126.1	Detailed Description	960
6.126.2	Member Typedef Documentation	961
6.127	std::_fwdlst::_Node< _ValPtr > Struct Template Reference	961
6.127.1	Detailed Description	962
6.128	__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference	962
6.128.1	Detailed Description	963
6.129	std::_fwdlst::_Node_base< _VoidPtr > Struct Template Reference	963
6.129.1	Detailed Description	963
6.130	__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc > Class Template Reference	964
6.130.1	Detailed Description	965
6.130.2	Member Typedef Documentation	965
6.130.3	Member Function Documentation	965
6.131	__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc > Class Template Reference	966
6.131.1	Detailed Description	968
6.131.2	Member Typedef Documentation	968
6.131.3	Member Function Documentation	968
6.132	__gnu_debug::_Not_equal_to< _Type > Class Template Reference	969
6.132.1	Detailed Description	969
6.133	std::_Not_fn< _Fn > Class Template Reference	970
6.133.1	Detailed Description	970
6.134	__gnu_parallel::_Nothing Struct Reference	970
6.134.1	Detailed Description	971
6.134.2	Member Function Documentation	971
6.135	__gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference	971
6.135.1	Detailed Description	971
6.135.2	Member Data Documentation	971
6.136	std::_Placeholder< _Num > Struct Template Reference	972

6.136.1 Detailed Description	972
6.137 <code>__gnu_parallel::Plus<_Tp1, _Tp2, _Result></code> Struct Template Reference	972
6.137.1 Detailed Description	973
6.137.2 Member Typedef Documentation	973
6.138 <code>__gnu_parallel::PMWMSortingData<_RAIter></code> Struct Template Reference	973
6.138.1 Detailed Description	974
6.138.2 Member Data Documentation	974
6.139 <code>__gnu_cxx::Pointer_adapter<_Storage_policy></code> Class Template Reference	975
6.139.1 Detailed Description	976
6.140 <code>__gnu_parallel::PseudoSequence<_Tp, _DifferenceTp></code> Class Template Reference	977
6.140.1 Detailed Description	977
6.140.2 Constructor & Destructor Documentation	977
6.140.3 Member Function Documentation	978
6.141 <code>__gnu_parallel::PseudoSequenceIterator<_Tp, _DifferenceTp></code> Class Template Reference	978
6.141.1 Detailed Description	978
6.142 <code>__gnu_parallel::QSBThreadLocal<_RAIter></code> Struct Template Reference	979
6.142.1 Detailed Description	979
6.142.2 Member Typedef Documentation	979
6.142.3 Constructor & Destructor Documentation	979
6.142.4 Member Data Documentation	980
6.143 <code>std::__detail::Quoted_string<_String, _CharT></code> Struct Template Reference	980
6.143.1 Detailed Description	981
6.144 <code>__gnu_parallel::RandomNumber</code> Class Reference	981
6.144.1 Detailed Description	981
6.144.2 Constructor & Destructor Documentation	981
6.144.3 Member Function Documentation	982
6.145 <code>__gnu_cxx::Relative_pointer_impl<_Tp></code> Class Template Reference	982
6.145.1 Detailed Description	983
6.146 <code>__gnu_cxx::Relative_pointer_impl<const _Tp></code> Class Template Reference	984
6.146.1 Detailed Description	984
6.147 <code>__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp></code> Class Template Reference	984
6.147.1 Detailed Description	985
6.147.2 Constructor & Destructor Documentation	985
6.147.3 Member Function Documentation	985
6.148 <code>__gnu_debug::Safe_container<_SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware></code> Class Template Reference	986
6.148.1 Detailed Description	988
6.149 <code>__gnu_debug::Safe_forward_list<_SafeSequence></code> Class Template Reference	988
6.149.1 Detailed Description	988

6.149.2 Member Function Documentation	989
6.149.3 Member Data Documentation	989
6.150 __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category> Class Template Reference	990
6.150.1 Detailed Description	992
6.150.2 Constructor & Destructor Documentation	992
6.150.3 Member Function Documentation	993
6.150.4 Member Data Documentation	998
6.151 __gnu_debug::_Safe_iterator_base Class Reference	999
6.151.1 Detailed Description	1001
6.151.2 Constructor & Destructor Documentation	1001
6.151.3 Member Function Documentation	1002
6.151.4 Member Data Documentation	1003
6.152 __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer> Class Template Reference	1004
6.152.1 Detailed Description	1007
6.152.2 Constructor & Destructor Documentation	1007
6.152.3 Member Function Documentation	1008
6.152.4 Member Data Documentation	1013
6.153 __gnu_debug::_Safe_local_iterator_base Class Reference	1013
6.153.1 Detailed Description	1015
6.153.2 Constructor & Destructor Documentation	1015
6.153.3 Member Function Documentation	1016
6.153.4 Member Data Documentation	1018
6.154 __gnu_debug::_Safe_node_sequence<_Sequence> Class Template Reference	1018
6.154.1 Detailed Description	1020
6.154.2 Member Function Documentation	1020
6.154.3 Member Data Documentation	1021
6.155 __gnu_debug::_Safe_sequence<_Sequence> Class Template Reference	1021
6.155.1 Detailed Description	1022
6.155.2 Member Function Documentation	1023
6.155.3 Member Data Documentation	1024
6.156 __gnu_debug::_Safe_sequence_base Class Reference	1024
6.156.1 Detailed Description	1026
6.156.2 Constructor & Destructor Documentation	1026
6.156.3 Member Function Documentation	1026
6.156.4 Member Data Documentation	1027
6.157 __gnu_debug::_Safe_unordered_container<_Container> Class Template Reference	1027
6.157.1 Detailed Description	1028
6.157.2 Member Function Documentation	1029
6.157.3 Member Data Documentation	1030

6.158	__gnu_debug::__Safe_unordered_container_base Class Reference	1030
6.158.1	Detailed Description	1032
6.158.2	Constructor & Destructor Documentation	1032
6.158.3	Member Function Documentation	1032
6.158.4	Member Data Documentation	1033
6.159	__gnu_debug::__Safe_vector< _SafeSequence, _BaseSequence > Class Template Reference	1034
6.159.1	Detailed Description	1035
6.160	__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference	1035
6.160.1	Detailed Description	1035
6.161	__gnu_parallel::__SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference	1035
6.161.1	Detailed Description	1035
6.162	std::__detail::__Scanner< _CharT > Class Template Reference	1036
6.162.1	Detailed Description	1037
6.162.2	Member Enumeration Documentation	1037
6.163	__gnu_debug::__Sequence_traits< _Sequence > Struct Template Reference	1037
6.163.1	Detailed Description	1037
6.164	__gnu_parallel::__Settings Struct Reference	1037
6.164.1	Detailed Description	1039
6.164.2	Member Function Documentation	1039
6.164.3	Member Data Documentation	1039
6.165	std::__Sp_ebo_helper< _Tp, false > Struct Template Reference	1044
6.165.1	Detailed Description	1044
6.166	std::__Sp_ebo_helper< _Tp, true > Struct Template Reference	1044
6.166.1	Detailed Description	1045
6.167	__gnu_parallel::__SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	1045
6.167.1	Detailed Description	1045
6.168	__gnu_parallel::__SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	1045
6.168.1	Detailed Description	1045
6.169	__gnu_parallel::__SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference	1045
6.169.1	Detailed Description	1046
6.170	std::__detail::__StateSeq< _TraitsT > Class Template Reference	1046
6.170.1	Detailed Description	1046
6.171	__gnu_cxx::__Std_pointer_impl< _Tp > Class Template Reference	1046
6.171.1	Detailed Description	1047
6.172	__gnu_cxx::__Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference	1047
6.172.1	Detailed Description	1048
6.172.2	Constructor & Destructor Documentation	1048

6.172.3 Member Function Documentation	1048
6.173 <code>std::_Temporary_buffer<_ForwardIterator, _Tp></code> Class Template Reference	1048
6.173.1 Detailed Description	1049
6.173.2 Constructor & Destructor Documentation	1049
6.173.3 Member Function Documentation	1050
6.174 <code>__gnu_cxx::_Unqualified_type<_Tp></code> Struct Template Reference	1050
6.174.1 Detailed Description	1050
6.175 <code>std::_Vector_base<_Tp, _Alloc></code> Struct Template Reference	1051
6.175.1 Detailed Description	1052
6.176 <code>std::add_const<_Tp></code> Struct Template Reference	1052
6.176.1 Detailed Description	1053
6.177 <code>std::add_cv<_Tp></code> Struct Template Reference	1053
6.177.1 Detailed Description	1053
6.178 <code>std::add_lvalue_reference<_Tp></code> Struct Template Reference	1053
6.178.1 Detailed Description	1053
6.179 <code>std::add_pointer<_Tp></code> Struct Template Reference	1053
6.179.1 Detailed Description	1054
6.180 <code>std::add_rvalue_reference<_Tp></code> Struct Template Reference	1054
6.180.1 Detailed Description	1054
6.181 <code>std::add_volatile<_Tp></code> Struct Template Reference	1054
6.181.1 Detailed Description	1054
6.182 <code>std::adopt_lock_t</code> Struct Reference	1054
6.182.1 Detailed Description	1054
6.183 <code>std::aligned_storage<_Len, _Align></code> Struct Template Reference	1055
6.183.1 Detailed Description	1055
6.184 <code>std::aligned_union<_Len, _Types></code> Struct Template Reference	1055
6.184.1 Detailed Description	1055
6.185 <code>std::alignment_of<_Tp></code> Struct Template Reference	1056
6.185.1 Detailed Description	1056
6.186 <code>__gnu_cxx::allocator<_Tp></code> Class Template Reference	1056
6.186.1 Detailed Description	1058
6.187 <code>std::allocator<_Tp></code> Class Template Reference	1058
6.187.1 Detailed Description	1059
6.188 <code>std::allocator<void></code> Class Reference	1059
6.188.1 Detailed Description	1060
6.188.2 Friends And Related Symbol Documentation	1060
6.189 <code>std::allocator_traits<_Alloc></code> Struct Template Reference	1061
6.189.1 Detailed Description	1063
6.189.2 Member Typedef Documentation	1063

6.189.3 Member Function Documentation	1065
6.190 std::allocator_traits< allocator< _Tp > > Struct Template Reference	1068
6.190.1 Detailed Description	1069
6.190.2 Member Typedef Documentation	1069
6.190.3 Member Function Documentation	1070
6.191 std::allocator_traits< allocator< void > > Struct Reference	1073
6.191.1 Detailed Description	1074
6.191.2 Member Typedef Documentation	1074
6.191.3 Member Function Documentation	1075
6.192 std::allocator_traits< pmr::polymorphic_allocator< _Tp > > Struct Template Reference	1077
6.192.1 Detailed Description	1078
6.192.2 Member Typedef Documentation	1078
6.192.3 Member Function Documentation	1079
6.193 __gnu_cxx::limit_condition::always_adjustor Struct Reference	1082
6.193.1 Detailed Description	1082
6.194 __gnu_cxx::random_condition::always_adjustor Struct Reference	1082
6.194.1 Detailed Description	1082
6.195 __gnu_cxx::annotate_base Struct Reference	1082
6.195.1 Detailed Description	1083
6.196 std::experimental::fundamentals_v1::any Class Reference	1083
6.196.1 Detailed Description	1084
6.196.2 Constructor & Destructor Documentation	1084
6.196.3 Member Function Documentation	1085
6.197 std::array< _Tp, _Nm > Struct Template Reference	1086
6.197.1 Detailed Description	1087
6.198 __gnu_pbds::associative_tag Struct Reference	1088
6.198.1 Detailed Description	1088
6.199 std::atomic< _Tp > Struct Template Reference	1088
6.199.1 Detailed Description	1089
6.200 std::atomic< _Tp * > Struct Template Reference	1089
6.200.1 Detailed Description	1090
6.201 std::atomic< bool > Struct Reference	1090
6.201.1 Detailed Description	1091
6.202 std::atomic< char > Struct Reference	1091
6.202.1 Detailed Description	1092
6.203 std::atomic< char16_t > Struct Reference	1092
6.203.1 Detailed Description	1092
6.204 std::atomic< char32_t > Struct Reference	1093
6.204.1 Detailed Description	1093

6.205 <code>std::atomic< int ></code> Struct Reference	1093
6.205.1 Detailed Description	1094
6.206 <code>std::atomic< long ></code> Struct Reference	1094
6.206.1 Detailed Description	1094
6.207 <code>std::atomic< long long ></code> Struct Reference	1094
6.207.1 Detailed Description	1095
6.208 <code>std::atomic< short ></code> Struct Reference	1095
6.208.1 Detailed Description	1095
6.209 <code>std::atomic< signed char ></code> Struct Reference	1096
6.209.1 Detailed Description	1096
6.210 <code>std::atomic< unsigned char ></code> Struct Reference	1096
6.210.1 Detailed Description	1097
6.211 <code>std::atomic< unsigned int ></code> Struct Reference	1097
6.211.1 Detailed Description	1097
6.212 <code>std::atomic< unsigned long ></code> Struct Reference	1097
6.212.1 Detailed Description	1098
6.213 <code>std::atomic< unsigned long long ></code> Struct Reference	1098
6.213.1 Detailed Description	1099
6.214 <code>std::atomic< unsigned short ></code> Struct Reference	1099
6.214.1 Detailed Description	1099
6.215 <code>std::atomic< wchar_t ></code> Struct Reference	1099
6.215.1 Detailed Description	1100
6.216 <code>std::atomic_flag</code> Struct Reference	1100
6.216.1 Detailed Description	1100
6.217 <code>std::auto_ptr< _Tp ></code> Class Template Reference	1100
6.217.1 Detailed Description	1101
6.217.2 Member Typedef Documentation	1101
6.217.3 Constructor & Destructor Documentation	1102
6.217.4 Member Function Documentation	1103
6.218 <code>std::auto_ptr_ref< _Tp1 ></code> Struct Template Reference	1105
6.218.1 Detailed Description	1105
6.219 <code>std::back_insert_iterator< _Container ></code> Class Template Reference	1105
6.219.1 Detailed Description	1106
6.219.2 Member Typedef Documentation	1107
6.219.3 Constructor & Destructor Documentation	1107
6.219.4 Member Function Documentation	1107
6.220 <code>std::bad_alloc</code> Class Reference	1108
6.220.1 Detailed Description	1109
6.220.2 Member Function Documentation	1109

6.221	std::experimental::fundamentals_v1::bad_any_cast Class Reference	1109
6.221.1	Detailed Description	1109
6.221.2	Member Function Documentation	1110
6.222	std::bad_cast Class Reference	1110
6.222.1	Detailed Description	1110
6.222.2	Member Function Documentation	1110
6.223	std::bad_exception Class Reference	1111
6.223.1	Detailed Description	1111
6.223.2	Member Function Documentation	1111
6.224	std::bad_function_call Class Reference	1111
6.224.1	Detailed Description	1112
6.224.2	Member Function Documentation	1112
6.225	std::experimental::fundamentals_v1::bad_optional_access Class Reference	1112
6.225.1	Detailed Description	1113
6.225.2	Member Function Documentation	1113
6.226	std::bad_typeid Class Reference	1113
6.226.1	Detailed Description	1114
6.226.2	Member Function Documentation	1114
6.227	std::bad_weak_ptr Class Reference	1114
6.227.1	Detailed Description	1115
6.227.2	Member Function Documentation	1115
6.228	__gnu_parallel::balanced_quicksort_tag Struct Reference	1115
6.228.1	Detailed Description	1116
6.228.2	Member Function Documentation	1116
6.229	__gnu_parallel::balanced_tag Struct Reference	1117
6.229.1	Detailed Description	1117
6.229.2	Member Function Documentation	1117
6.230	std::tr2::bases< _Tp > Struct Template Reference	1118
6.230.1	Detailed Description	1118
6.231	__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference	1118
6.231.1	Detailed Description	1119
6.232	__gnu_pbds::basic_branch_tag Struct Reference	1119
6.232.1	Detailed Description	1119
6.233	std::basic_filebuf< _CharT, _Traits > Class Template Reference	1119
6.233.1	Detailed Description	1123
6.233.2	Constructor & Destructor Documentation	1123
6.233.3	Member Function Documentation	1123
6.233.4	Member Data Documentation	1139

6.234 <code>std::basic_fstream<_CharT, _Traits></code> Class Template Reference	1141
6.234.1 Detailed Description	1148
6.234.2 Member Typedef Documentation	1148
6.234.3 Member Enumeration Documentation	1150
6.234.4 Constructor & Destructor Documentation	1150
6.234.5 Member Function Documentation	1151
6.234.6 Member Data Documentation	1186
6.235 <code>__gnu_pbds::basic_hash_table<Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc></code> Class Template Reference	1191
6.235.1 Detailed Description	1192
6.236 <code>__gnu_pbds::basic_hash_tag</code> Struct Reference	1193
6.236.1 Detailed Description	1193
6.237 <code>std::basic_ifstream<_CharT, _Traits></code> Class Template Reference	1194
6.237.1 Detailed Description	1199
6.237.2 Member Typedef Documentation	1199
6.237.3 Member Enumeration Documentation	1201
6.237.4 Constructor & Destructor Documentation	1201
6.237.5 Member Function Documentation	1202
6.237.6 Member Data Documentation	1230
6.238 <code>__gnu_pbds::basic_invalidation_guarantee</code> Struct Reference	1234
6.238.1 Detailed Description	1235
6.239 <code>std::basic_ios<_CharT, _Traits></code> Class Template Reference	1235
6.239.1 Detailed Description	1239
6.239.2 Member Typedef Documentation	1239
6.239.3 Member Enumeration Documentation	1242
6.239.4 Constructor & Destructor Documentation	1242
6.239.5 Member Function Documentation	1243
6.239.6 Member Data Documentation	1254
6.240 <code>std::basic_iostream<_CharT, _Traits></code> Class Template Reference	1259
6.240.1 Detailed Description	1265
6.240.2 Member Typedef Documentation	1265
6.240.3 Member Enumeration Documentation	1267
6.240.4 Constructor & Destructor Documentation	1267
6.240.5 Member Function Documentation	1267
6.240.6 Member Data Documentation	1301
6.241 <code>std::basic_istream<_CharT, _Traits></code> Class Template Reference	1306
6.241.1 Detailed Description	1311
6.241.2 Member Typedef Documentation	1311
6.241.3 Member Enumeration Documentation	1313

6.241.4 Constructor & Destructor Documentation	1314
6.241.5 Member Function Documentation	1314
6.241.6 Member Data Documentation	1340
6.242 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference	1345
6.242.1 Detailed Description	1351
6.242.2 Member Typedef Documentation	1351
6.242.3 Member Enumeration Documentation	1353
6.242.4 Constructor & Destructor Documentation	1354
6.242.5 Member Function Documentation	1354
6.242.6 Member Data Documentation	1381
6.243 std::basic_ofstream< _CharT, _Traits > Class Template Reference	1386
6.243.1 Detailed Description	1391
6.243.2 Member Typedef Documentation	1391
6.243.3 Member Enumeration Documentation	1393
6.243.4 Constructor & Destructor Documentation	1393
6.243.5 Member Function Documentation	1394
6.243.6 Member Data Documentation	1415
6.244 std::basic_ostream< _CharT, _Traits > Class Template Reference	1419
6.244.1 Detailed Description	1424
6.244.2 Member Typedef Documentation	1424
6.244.3 Member Enumeration Documentation	1426
6.244.4 Constructor & Destructor Documentation	1427
6.244.5 Member Function Documentation	1427
6.244.6 Member Data Documentation	1446
6.245 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	1451
6.245.1 Detailed Description	1456
6.245.2 Member Typedef Documentation	1457
6.245.3 Member Enumeration Documentation	1459
6.245.4 Constructor & Destructor Documentation	1459
6.245.5 Member Function Documentation	1460
6.245.6 Member Data Documentation	1479
6.246 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	1484
6.246.1 Detailed Description	1486
6.246.2 Constructor & Destructor Documentation	1486
6.246.3 Member Function Documentation	1489
6.247 std::basic_streambuf< _CharT, _Traits > Class Template Reference	1494
6.247.1 Detailed Description	1496
6.247.2 Member Typedef Documentation	1497
6.247.3 Constructor & Destructor Documentation	1498

6.247.4 Member Function Documentation	1498
6.247.5 Member Data Documentation	1511
6.248 <code>__gnu_debug::basic_string<_CharT, _Traits, _Allocator></code> Class Template Reference	1512
6.248.1 Detailed Description	1519
6.248.2 Member Function Documentation	1519
6.248.3 Member Data Documentation	1589
6.249 <code>std::basic_string<_CharT, _Traits, _Alloc></code> Class Template Reference	1589
6.249.1 Detailed Description	1598
6.249.2 Constructor & Destructor Documentation	1599
6.249.3 Member Function Documentation	1606
6.249.4 Member Data Documentation	1694
6.250 <code>std::basic_string_view<_CharT, _Traits></code> Class Template Reference	1695
6.250.1 Detailed Description	1697
6.251 <code>std::experimental::filesystem::v1::basic_string_view<_CharT, _Traits></code> Class Template Reference	1697
6.251.1 Detailed Description	1699
6.252 <code>std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits></code> Class Template Reference	1699
6.252.1 Detailed Description	1701
6.253 <code>std::basic_stringbuf<_CharT, _Traits, _Alloc></code> Class Template Reference	1701
6.253.1 Detailed Description	1704
6.253.2 Constructor & Destructor Documentation	1704
6.253.3 Member Function Documentation	1705
6.253.4 Member Data Documentation	1719
6.254 <code>std::basic_stringstream<_CharT, _Traits, _Alloc></code> Class Template Reference	1720
6.254.1 Detailed Description	1726
6.254.2 Member Typedef Documentation	1727
6.254.3 Member Enumeration Documentation	1728
6.254.4 Constructor & Destructor Documentation	1729
6.254.5 Member Function Documentation	1729
6.254.6 Member Data Documentation	1764
6.255 <code>std::bernoulli_distribution</code> Class Reference	1768
6.255.1 Detailed Description	1769
6.255.2 Member Typedef Documentation	1769
6.255.3 Constructor & Destructor Documentation	1769
6.255.4 Member Function Documentation	1770
6.255.5 Friends And Related Symbol Documentation	1771
6.256 <code>std::bidirectional_iterator_tag</code> Struct Reference	1771
6.256.1 Detailed Description	1772
6.257 <code>__gnu_pbds::detail::bin_search_tree_const_it<Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc></code> Class Template Reference	1772

6.257.1 Detailed Description	1773
6.258 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	1774
6.258.1 Detailed Description	1774
6.258.2 Member Typedef Documentation	1775
6.258.3 Member Function Documentation	1775
6.259 __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference	1776
6.259.1 Detailed Description	1778
6.260 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	1778
6.260.1 Detailed Description	1779
6.260.2 Member Typedef Documentation	1779
6.260.3 Member Function Documentation	1780
6.261 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference	1781
6.261.1 Detailed Description	1781
6.261.2 Member Typedef Documentation	1782
6.262 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference	1782
6.262.1 Detailed Description	1782
6.262.2 Member Typedef Documentation	1782
6.263 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference	1783
6.263.1 Detailed Description	1783
6.263.2 Member Typedef Documentation	1783
6.264 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference	1784
6.264.1 Detailed Description	1786
6.264.2 Member Typedef Documentation	1786
6.265 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	1786
6.265.1 Detailed Description	1788
6.266 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference	1788
6.266.1 Detailed Description	1789
6.266.2 Member Typedef Documentation	1789
6.266.3 Constructor & Destructor Documentation	1790
6.266.4 Member Function Documentation	1790
6.267 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference	1791
6.267.1 Detailed Description	1793
6.267.2 Member Typedef Documentation	1793
6.267.3 Constructor & Destructor Documentation	1794

6.267.4 Member Function Documentation	1794
6.268 <code>__gnu_pbds::binary_heap_tag</code> Struct Reference	1795
6.268.1 Detailed Description	1795
6.269 <code>std::binary_negate<_Predicate></code> Class Template Reference	1795
6.269.1 Detailed Description	1796
6.269.2 Member Typedef Documentation	1796
6.270 <code>std::binder1st<_Operation></code> Class Template Reference	1796
6.270.1 Detailed Description	1797
6.270.2 Member Typedef Documentation	1797
6.271 <code>std::binder2nd<_Operation></code> Class Template Reference	1798
6.271.1 Detailed Description	1798
6.271.2 Member Typedef Documentation	1799
6.272 <code>std::binomial_distribution<_IntType></code> Class Template Reference	1799
6.272.1 Detailed Description	1800
6.272.2 Member Typedef Documentation	1800
6.272.3 Member Function Documentation	1800
6.272.4 Friends And Related Symbol Documentation	1801
6.273 <code>__gnu_pbds::detail::binomial_heap<Value_Type, Cmp_Fn, _Alloc></code> Class Template Reference	1802
6.273.1 Detailed Description	1804
6.274 <code>__gnu_pbds::detail::binomial_heap_base<Value_Type, Cmp_Fn, _Alloc></code> Class Template Reference	1804
6.274.1 Detailed Description	1806
6.275 <code>__gnu_pbds::binomial_heap_tag</code> Struct Reference	1807
6.275.1 Detailed Description	1807
6.276 <code>__gnu_cxx::bitmap_allocator<_Tp></code> Class Template Reference	1807
6.276.1 Detailed Description	1808
6.276.2 Member Function Documentation	1808
6.277 <code>std::__debug::bitset<_Nb></code> Class Template Reference	1809
6.277.1 Detailed Description	1810
6.278 <code>std::bitset<_Nb></code> Class Template Reference	1810
6.278.1 Detailed Description	1812
6.278.2 Constructor & Destructor Documentation	1813
6.278.3 Member Function Documentation	1815
6.279 <code>std::tr2::bool_set</code> Class Reference	1821
6.279.1 Detailed Description	1821
6.279.2 Constructor & Destructor Documentation	1822
6.279.3 Member Function Documentation	1822
6.280 <code>__gnu_pbds::detail::branch_policy<Node_Cltr, Node_Itr, _Alloc></code> Struct Template Reference	1822
6.280.1 Detailed Description	1823
6.281 <code>__gnu_pbds::detail::branch_policy<Node_Cltr, Node_Cltr, _Alloc></code> Struct Template Reference	1823

6.281.1 Detailed Description	1824
6.282 <code>std::cauchy_distribution<_RealType></code> Class Template Reference	1824
6.282.1 Detailed Description	1825
6.282.2 Member Typedef Documentation	1825
6.282.3 Member Function Documentation	1825
6.282.4 Friends And Related Symbol Documentation	1826
6.283 <code>__gnu_pbds::cc_hash_max_collision_check_resize_trigger<External_Load_Access, Size_Type></code> Class Template Reference	1826
6.283.1 Detailed Description	1827
6.283.2 Member Enumeration Documentation	1827
6.283.3 Constructor & Destructor Documentation	1828
6.283.4 Member Function Documentation	1828
6.284 <code>__gnu_pbds::cc_hash_table<Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc></code> Class Template Reference	1830
6.284.1 Detailed Description	1832
6.284.2 Constructor & Destructor Documentation	1832
6.285 <code>__gnu_pbds::cc_hash_tag</code> Struct Reference	1835
6.285.1 Detailed Description	1836
6.286 <code>__gnu_pbds::detail::cc_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy></code> Class Template Reference	1836
6.286.1 Detailed Description	1839
6.286.2 Member Enumeration Documentation	1839
6.286.3 Member Function Documentation	1839
6.287 <code>__gnu_cxx::char_traits<_CharT></code> Struct Template Reference	1841
6.287.1 Detailed Description	1842
6.288 <code>std::char_traits<_CharT></code> Struct Template Reference	1842
6.288.1 Detailed Description	1843
6.289 <code>std::char_traits<__gnu_cxx::character<_Value, _Int, _St>></code> Struct Template Reference	1843
6.289.1 Detailed Description	1845
6.290 <code>std::char_traits<char></code> Struct Reference	1845
6.290.1 Detailed Description	1847
6.291 <code>std::char_traits<wchar_t></code> Struct Reference	1847
6.291.1 Detailed Description	1849
6.292 <code>__gnu_cxx::character<_Value, _Int, _St></code> Struct Template Reference	1849
6.292.1 Detailed Description	1849
6.293 <code>std::chi_squared_distribution<_RealType></code> Class Template Reference	1849
6.293.1 Detailed Description	1850
6.293.2 Member Typedef Documentation	1851
6.293.3 Member Function Documentation	1851
6.293.4 Friends And Related Symbol Documentation	1852

6.294 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	1853
6.294.1 Detailed Description	1854
6.294.2 Member Function Documentation	1854
6.295 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	1858
6.295.1 Detailed Description	1860
6.295.2 Member Function Documentation	1860
6.296 std::codecvt< char, char, mbstate_t > Class Reference	1866
6.296.1 Detailed Description	1867
6.296.2 Member Function Documentation	1867
6.297 std::codecvt< char16_t, char, mbstate_t > Class Reference	1873
6.297.1 Detailed Description	1875
6.297.2 Member Function Documentation	1875
6.298 std::codecvt< char32_t, char, mbstate_t > Class Reference	1880
6.298.1 Detailed Description	1882
6.298.2 Member Function Documentation	1882
6.299 std::codecvt< wchar_t, char, mbstate_t > Class Reference	1888
6.299.1 Detailed Description	1890
6.299.2 Member Function Documentation	1890
6.300 std::codecvt_base Class Reference	1895
6.300.1 Detailed Description	1896
6.301 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	1896
6.301.1 Detailed Description	1898
6.301.2 Member Function Documentation	1898
6.302 std::__cxx11::collate< _CharT > Class Template Reference	1902
6.302.1 Detailed Description	1903
6.302.2 Member Typedef Documentation	1903
6.302.3 Constructor & Destructor Documentation	1903
6.302.4 Member Function Documentation	1904
6.302.5 Member Data Documentation	1907
6.303 std::__cxx11::collate_byname< _CharT > Class Template Reference	1907
6.303.1 Detailed Description	1908
6.303.2 Member Typedef Documentation	1908
6.304 std::common_iterator< _It, _Sent > Class Template Reference	1908
6.304.1 Detailed Description	1909
6.305 std::common_type< _Tp > Struct Template Reference	1909
6.305.1 Detailed Description	1909
6.306 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference	1909
6.306.1 Detailed Description	1909

6.307 <code>std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference</code>	1910
6.307.1 Detailed Description	1910
6.308 <code>std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference</code>	1910
6.308.1 Detailed Description	1910
6.309 <code>std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference</code>	1910
6.309.1 Detailed Description	1910
6.310 <code>std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference</code>	1910
6.310.1 Detailed Description	1911
6.311 <code>std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > > Struct Template Reference</code>	1911
6.311.1 Detailed Description	1911
6.312 <code>std::compare_three_way_result< _Tp, _Up > Struct Template Reference</code>	1911
6.312.1 Detailed Description	1911
6.313 <code>std::complex< _Tp > Class Template Reference</code>	1911
6.313.1 Detailed Description	1912
6.313.2 Member Typedef Documentation	1912
6.313.3 Constructor & Destructor Documentation	1912
6.313.4 Member Function Documentation	1913
6.314 <code>std::complex< double > Class Reference</code>	1914
6.314.1 Detailed Description	1915
6.315 <code>std::complex< float > Class Reference</code>	1915
6.315.1 Detailed Description	1916
6.316 <code>std::complex< long double > Class Reference</code>	1916
6.316.1 Detailed Description	1917
6.317 <code>__gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference</code>	1917
6.317.1 Detailed Description	1917
6.318 <code>__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference</code>	1918
6.318.1 Detailed Description	1918
6.319 <code>__gnu_cxx::condition_base Struct Reference</code>	1918
6.319.1 Detailed Description	1919
6.320 <code>std::condition_variable Class Reference</code>	1919
6.320.1 Detailed Description	1919
6.321 <code>std::condition_variable_any Class Reference</code>	1919
6.321.1 Detailed Description	1920
6.322 <code>std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference</code>	1920
6.322.1 Detailed Description	1920

6.323 __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::const_iterator Struct Reference	1920
6.323.1 Detailed Description	1921
6.324 std::chrono::tzdb_list::const_iterator Class Reference	1922
6.324.1 Detailed Description	1922
6.325 std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg > Class Template Reference	1922
6.325.1 Detailed Description	1923
6.325.2 Member Typedef Documentation	1923
6.326 std::const_mem_fun1_t<_Ret, _Tp, _Arg > Class Template Reference	1924
6.326.1 Detailed Description	1924
6.326.2 Member Typedef Documentation	1924
6.327 std::const_mem_fun_ref_t<_Ret, _Tp > Class Template Reference	1925
6.327.1 Detailed Description	1925
6.327.2 Member Typedef Documentation	1925
6.328 std::const_mem_fun_t<_Ret, _Tp > Class Template Reference	1926
6.328.1 Detailed Description	1926
6.328.2 Member Typedef Documentation	1926
6.329 __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2 > Struct Template Reference	1927
6.329.1 Detailed Description	1927
6.330 __gnu_parallel::constant_size_blocks_tag Struct Reference	1927
6.330.1 Detailed Description	1928
6.331 __gnu_cxx::constant_unary_fun<_Result, _Argument > Struct Template Reference	1928
6.331.1 Detailed Description	1928
6.332 __gnu_cxx::constant_void_fun<_Result > Struct Template Reference	1929
6.332.1 Detailed Description	1929
6.333 __gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_TI > Struct Template Reference	1929
6.333.1 Detailed Description	1929
6.334 __gnu_pbds::detail::container_base_dispatch<_VTP, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference	1929
6.334.1 Detailed Description	1929
6.334.2 Member Typedef Documentation	1930
6.335 __gnu_pbds::detail::container_base_dispatch<_VTP, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference	1930
6.335.1 Detailed Description	1930
6.335.2 Member Typedef Documentation	1930
6.336 __gnu_pbds::detail::container_base_dispatch<_VTP, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference	1930
6.336.1 Detailed Description	1930
6.336.2 Member Typedef Documentation	1931

6.337	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference	1931
6.337.1	Detailed Description	1931
6.337.2	Member Typedef Documentation	1931
6.338	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference	1931
6.338.1	Detailed Description	1931
6.338.2	Member Typedef Documentation	1932
6.339	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	1932
6.339.1	Detailed Description	1932
6.339.2	Member Typedef Documentation	1932
6.340	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference	1932
6.340.1	Detailed Description	1932
6.340.2	Member Typedef Documentation	1933
6.341	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI > Struct Template Reference	1933
6.341.1	Detailed Description	1933
6.341.2	Member Typedef Documentation	1933
6.342	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI > Struct Template Reference	1933
6.342.1	Detailed Description	1933
6.342.2	Member Typedef Documentation	1934
6.343	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI > Struct Template Reference	1934
6.343.1	Detailed Description	1934
6.344	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI > Struct Template Reference	1934
6.344.1	Detailed Description	1934
6.344.2	Member Typedef Documentation	1934
6.345	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI > Struct Template Reference	1935
6.345.1	Detailed Description	1935
6.345.2	Member Typedef Documentation	1935
6.346	__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	1935
6.346.1	Detailed Description	1935
6.346.2	Member Typedef Documentation	1935
6.347	__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference	1936
6.347.1	Detailed Description	1936

6.347.2 Member Typedef Documentation	1936
6.348 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI ></code> Struct Template Reference	1936
6.348.1 Detailed Description	1936
6.348.2 Member Typedef Documentation	1936
6.349 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI > Struct</code> Template Reference	1937
6.349.1 Detailed Description	1937
6.349.2 Member Typedef Documentation	1937
6.350 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI > Struct</code> Template Reference	1937
6.350.1 Detailed Description	1937
6.350.2 Member Typedef Documentation	1937
6.351 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI > Struct</code> Template Reference	1938
6.351.1 Detailed Description	1938
6.352 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI ></code> Struct Template Reference	1938
6.352.1 Detailed Description	1938
6.352.2 Member Typedef Documentation	1938
6.353 <code>__gnu_pbds::container_error</code> Struct Reference	1938
6.353.1 Detailed Description	1939
6.353.2 Member Function Documentation	1939
6.354 <code>__gnu_pbds::container_tag</code> Struct Reference	1939
6.354.1 Detailed Description	1940
6.355 <code>__gnu_pbds::container_traits< Cntnr ></code> Struct Template Reference	1940
6.355.1 Detailed Description	1941
6.355.2 Member Enumeration Documentation	1941
6.356 <code>__gnu_pbds::container_traits_base< _Tag ></code> Struct Template Reference	1941
6.356.1 Detailed Description	1941
6.357 <code>__gnu_pbds::container_traits_base< binary_heap_tag ></code> Struct Reference	1941
6.357.1 Detailed Description	1942
6.358 <code>__gnu_pbds::container_traits_base< binomial_heap_tag ></code> Struct Reference	1942
6.358.1 Detailed Description	1942
6.359 <code>__gnu_pbds::container_traits_base< cc_hash_tag ></code> Struct Reference	1942
6.359.1 Detailed Description	1942
6.360 <code>__gnu_pbds::container_traits_base< gp_hash_tag ></code> Struct Reference	1942
6.360.1 Detailed Description	1942
6.361 <code>__gnu_pbds::container_traits_base< list_update_tag ></code> Struct Reference	1943
6.361.1 Detailed Description	1943

6.362	__gnu_pbds::container_traits_base< ov_tree_tag > Struct Reference	1943
6.362.1	Detailed Description	1943
6.363	__gnu_pbds::container_traits_base< pairing_heap_tag > Struct Reference	1943
6.363.1	Detailed Description	1943
6.364	__gnu_pbds::container_traits_base< pat_trie_tag > Struct Reference	1943
6.364.1	Detailed Description	1944
6.365	__gnu_pbds::container_traits_base< rb_tree_tag > Struct Reference	1944
6.365.1	Detailed Description	1944
6.366	__gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Reference	1944
6.366.1	Detailed Description	1944
6.367	__gnu_pbds::container_traits_base< splay_tree_tag > Struct Reference	1944
6.367.1	Detailed Description	1945
6.368	__gnu_pbds::container_traits_base< thin_heap_tag > Struct Reference	1945
6.368.1	Detailed Description	1945
6.369	std::contiguous_iterator_tag Struct Reference	1945
6.369.1	Detailed Description	1946
6.370	std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference	1946
6.370.1	Detailed Description	1947
6.370.2	Constructor & Destructor Documentation	1947
6.370.3	Member Function Documentation	1949
6.370.4	Friends And Related Symbol Documentation	1950
6.371	std::counted_iterator< _It > Class Template Reference	1950
6.371.1	Detailed Description	1951
6.372	std::ctype< _CharT > Class Template Reference	1952
6.372.1	Detailed Description	1953
6.372.2	Member Function Documentation	1954
6.372.3	Member Data Documentation	1965
6.373	std::ctype< char > Class Reference	1965
6.373.1	Detailed Description	1968
6.373.2	Member Typedef Documentation	1968
6.373.3	Constructor & Destructor Documentation	1969
6.373.4	Member Function Documentation	1969
6.373.5	Member Data Documentation	1995
6.374	std::ctype< wchar_t > Class Reference	1996
6.374.1	Detailed Description	1998
6.374.2	Member Typedef Documentation	1998
6.374.3	Constructor & Destructor Documentation	1998
6.374.4	Member Function Documentation	1999

6.374.5 Member Data Documentation	2009
6.375 std::ctype_base Struct Reference	2009
6.375.1 Detailed Description	2010
6.376 std::ctype_byname< _CharT > Class Template Reference	2010
6.376.1 Detailed Description	2012
6.376.2 Member Function Documentation	2013
6.376.3 Member Data Documentation	2024
6.377 std::ctype_byname< char > Class Reference	2024
6.377.1 Detailed Description	2027
6.377.2 Member Function Documentation	2027
6.377.3 Member Data Documentation	2047
6.378 std::ranges::dangling Struct Reference	2048
6.378.1 Detailed Description	2048
6.379 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference	2048
6.379.1 Detailed Description	2049
6.380 std::decimal::decimal128 Class Reference	2049
6.380.1 Detailed Description	2050
6.380.2 Constructor & Destructor Documentation	2050
6.381 std::decimal::decimal32 Class Reference	2050
6.381.1 Detailed Description	2052
6.381.2 Constructor & Destructor Documentation	2052
6.382 std::decimal::decimal64 Class Reference	2052
6.382.1 Detailed Description	2053
6.382.2 Constructor & Destructor Documentation	2053
6.383 parallelism_v2::simd_abi::deduce< _Tp, _Np,... > Struct Template Reference	2053
6.383.1 Detailed Description	2054
6.384 __gnu_pbds::detail::default_comb_hash_fn Struct Reference	2054
6.384.1 Detailed Description	2054
6.384.2 Member Typedef Documentation	2054
6.385 std::default_delete< _Tp > Struct Template Reference	2054
6.385.1 Detailed Description	2054
6.385.2 Constructor & Destructor Documentation	2055
6.385.3 Member Function Documentation	2055
6.386 std::default_delete< _Tp[]> Struct Template Reference	2055
6.386.1 Detailed Description	2055
6.386.2 Constructor & Destructor Documentation	2056
6.386.3 Member Function Documentation	2056
6.387 __gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference	2056
6.387.1 Detailed Description	2056

6.387.2 Member Typedef Documentation	2056
6.388 __gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference	2057
6.388.1 Detailed Description	2057
6.388.2 Member Typedef Documentation	2057
6.389 __gnu_parallel::default_parallel_tag Struct Reference	2057
6.389.1 Detailed Description	2058
6.389.2 Member Function Documentation	2058
6.390 __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference	2058
6.390.1 Detailed Description	2058
6.390.2 Member Typedef Documentation	2058
6.391 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference	2059
6.391.1 Detailed Description	2059
6.391.2 Member Typedef Documentation	2059
6.392 std::default_sentinel_t Struct Reference	2059
6.392.1 Detailed Description	2059
6.393 __gnu_pbds::detail::default_trie_access_traits< Key > Struct Template Reference	2059
6.393.1 Detailed Description	2059
6.394 __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference	2060
6.394.1 Detailed Description	2060
6.394.2 Member Typedef Documentation	2060
6.395 __gnu_pbds::detail::default_update_policy Struct Reference	2060
6.395.1 Detailed Description	2060
6.395.2 Member Typedef Documentation	2060
6.396 std::defer_lock_t Struct Reference	2060
6.396.1 Detailed Description	2061
6.397 std::__debug::deque< _Tp, _Allocator > Class Template Reference	2061
6.397.1 Detailed Description	2063
6.398 std::deque< _Tp, _Alloc > Class Template Reference	2063
6.398.1 Detailed Description	2067
6.398.2 Constructor & Destructor Documentation	2068
6.398.3 Member Function Documentation	2071
6.399 std::destroying_delete_t Struct Reference	2085
6.399.1 Detailed Description	2085
6.400 std::tr2::direct_bases< _Tp > Struct Template Reference	2086
6.400.1 Detailed Description	2086
6.401 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference	2086
6.401.1 Detailed Description	2087
6.401.2 Member Function Documentation	2087

6.402 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference	2087
6.402.1 Detailed Description	2088
6.402.2 Member Function Documentation	2088
6.403 std::filesystem::directory_entry Class Reference	2088
6.403.1 Detailed Description	2089
6.404 std::filesystem::directory_iterator Class Reference	2089
6.404.1 Detailed Description	2090
6.405 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2091
6.405.1 Detailed Description	2091
6.405.2 Member Typedef Documentation	2092
6.405.3 Constructor & Destructor Documentation	2092
6.405.4 Member Function Documentation	2093
6.405.5 Friends And Related Symbol Documentation	2094
6.406 std::discrete_distribution< _IntType > Class Template Reference	2095
6.406.1 Detailed Description	2096
6.406.2 Member Typedef Documentation	2097
6.406.3 Member Function Documentation	2097
6.406.4 Friends And Related Symbol Documentation	2098
6.407 std::divides< _Tp > Struct Template Reference	2099
6.407.1 Detailed Description	2099
6.407.2 Member Typedef Documentation	2100
6.408 std::divides< void > Struct Reference	2100
6.408.1 Detailed Description	2101
6.408.2 Member Typedef Documentation	2101
6.409 std::domain_error Class Reference	2101
6.409.1 Detailed Description	2102
6.409.2 Member Function Documentation	2102
6.410 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference	2102
6.410.1 Detailed Description	2102
6.411 std::chrono::duration< _Rep, _Period > Class Template Reference	2102
6.411.1 Detailed Description	2103
6.412 std::chrono::duration_values< _Rep > Struct Template Reference	2104
6.412.1 Detailed Description	2104
6.413 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference	2104
6.413.1 Detailed Description	2106
6.413.2 Constructor & Destructor Documentation	2107
6.413.3 Member Function Documentation	2109
6.414 std::ranges::elements_of< _Range, _Alloc > Struct Template Reference	2118
6.414.1 Detailed Description	2118

6.415 <code>std::ranges::empty_view<_Tp></code> Class Template Reference	2118
6.415.1 Detailed Description	2119
6.416 <code>std::enable_if<bool, _Tp></code> Struct Template Reference	2119
6.416.1 Detailed Description	2119
6.417 <code>std::enable_shared_from_this<_Tp></code> Class Template Reference	2119
6.417.1 Detailed Description	2120
6.418 <code>__gnu_cxx::enc_filebuf<_CharT></code> Class Template Reference	2120
6.418.1 Detailed Description	2123
6.418.2 Member Function Documentation	2123
6.418.3 Member Data Documentation	2135
6.419 <code>__gnu_cxx::encoding_char_traits<_CharT></code> Struct Template Reference	2138
6.419.1 Detailed Description	2139
6.420 <code>__gnu_cxx::encoding_state</code> Class Reference	2139
6.420.1 Detailed Description	2140
6.421 <code>std::encoding_state</code> Class Reference	2140
6.421.1 Detailed Description	2141
6.422 <code>__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw></code> Struct Template Reference	2141
6.422.1 Detailed Description	2141
6.423 <code>__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false></code> Struct Template Reference	2141
6.423.1 Detailed Description	2141
6.424 <code>__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true></code> Struct Template Reference	2141
6.424.1 Detailed Description	2141
6.424.2 Member Typedef Documentation	2142
6.425 <code>__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw></code> Struct Template Reference	2142
6.425.1 Detailed Description	2142
6.426 <code>__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false></code> Struct Template Reference	2142
6.426.1 Detailed Description	2142
6.427 <code>__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true></code> Struct Template Reference	2142
6.427.1 Detailed Description	2142
6.428 <code>__gnu_pbds::detail::eq_by_less<Key, Cmp_Fn></code> Struct Template Reference	2143
6.428.1 Detailed Description	2143
6.429 <code>__gnu_parallel::equal_split_tag</code> Struct Reference	2143
6.429.1 Detailed Description	2143
6.430 <code>__gnu_cxx::equal_to<_Tp></code> Struct Template Reference	2144
6.430.1 Detailed Description	2144
6.430.2 Member Typedef Documentation	2144
6.431 <code>std::equal_to<_Tp></code> Struct Template Reference	2145
6.431.1 Detailed Description	2145
6.431.2 Member Typedef Documentation	2145

6.432 std::ranges::equal_to Struct Reference	2146
6.432.1 Detailed Description	2146
6.433 std::equal_to< void > Struct Reference	2146
6.433.1 Detailed Description	2147
6.433.2 Member Typedef Documentation	2147
6.434 std::error_category Class Reference	2147
6.434.1 Detailed Description	2148
6.434.2 Member Function Documentation	2148
6.435 std::error_code Class Reference	2149
6.435.1 Detailed Description	2150
6.435.2 Constructor & Destructor Documentation	2150
6.435.3 Member Function Documentation	2150
6.436 std::error_condition Class Reference	2151
6.436.1 Detailed Description	2151
6.436.2 Constructor & Destructor Documentation	2151
6.436.3 Member Function Documentation	2152
6.437 __gnu_parallel::exact_tag Struct Reference	2152
6.437.1 Detailed Description	2153
6.437.2 Member Function Documentation	2153
6.438 std::exception Class Reference	2154
6.438.1 Detailed Description	2154
6.438.2 Member Function Documentation	2155
6.439 std::__unspecified__::exception_ptr Class Reference	2155
6.439.1 Detailed Description	2155
6.440 std::exception_ptr Class Reference	2156
6.440.1 Detailed Description	2156
6.441 std::exponential_distribution< _RealType > Class Template Reference	2156
6.441.1 Detailed Description	2157
6.441.2 Member Typedef Documentation	2158
6.441.3 Constructor & Destructor Documentation	2158
6.441.4 Member Function Documentation	2158
6.441.5 Friends And Related Symbol Documentation	2159
6.442 std::extent< typename, _UInt > Struct Template Reference	2159
6.442.1 Detailed Description	2160
6.443 std::extreme_value_distribution< _RealType > Class Template Reference	2160
6.443.1 Detailed Description	2161
6.443.2 Member Typedef Documentation	2161
6.443.3 Member Function Documentation	2161
6.443.4 Friends And Related Symbol Documentation	2163

6.444	std::locale::facet Class Reference	2163
6.444.1	Detailed Description	2165
6.444.2	Constructor & Destructor Documentation	2165
6.445	std::ios_base::failure Class Reference	2166
6.445.1	Detailed Description	2166
6.445.2	Member Function Documentation	2166
6.446	std::filesystem::file_status Class Reference	2167
6.446.1	Detailed Description	2167
6.447	std::experimental::filesystem::v1::filesystem_error Class Reference	2167
6.447.1	Detailed Description	2168
6.447.2	Member Function Documentation	2168
6.448	std::filesystem::filesystem_error Class Reference	2169
6.448.1	Detailed Description	2169
6.448.2	Member Function Documentation	2170
6.449	__gnu_parallel::find_tag Struct Reference	2170
6.449.1	Detailed Description	2170
6.450	std::fisher_f_distribution<_RealType> Class Template Reference	2170
6.450.1	Detailed Description	2171
6.450.2	Member Typedef Documentation	2172
6.450.3	Member Function Documentation	2172
6.450.4	Friends And Related Symbol Documentation	2173
6.451	__gnu_cxx::forced_error Struct Reference	2174
6.451.1	Detailed Description	2174
6.451.2	Member Function Documentation	2174
6.452	std::forward_iterator_tag Struct Reference	2174
6.452.1	Detailed Description	2175
6.453	std::__debug::forward_list<_Tp, _Alloc> Class Template Reference	2175
6.453.1	Detailed Description	2178
6.454	std::forward_list<_Tp, _Alloc> Class Template Reference	2178
6.454.1	Detailed Description	2180
6.454.2	Constructor & Destructor Documentation	2180
6.454.3	Member Function Documentation	2183
6.455	std::fpos<_StateT> Class Template Reference	2196
6.455.1	Detailed Description	2196
6.455.2	Constructor & Destructor Documentation	2197
6.455.3	Member Function Documentation	2197
6.456	__gnu_cxx::free_list Class Reference	2198
6.456.1	Detailed Description	2198
6.456.2	Member Function Documentation	2199

6.457 <code>std::from_chars_result</code> Struct Reference	2199
6.457.1 Detailed Description	2200
6.458 <code>std::front_insert_iterator< _Container ></code> Class Template Reference	2200
6.458.1 Detailed Description	2201
6.458.2 Member Typedef Documentation	2201
6.458.3 Constructor & Destructor Documentation	2201
6.458.4 Member Function Documentation	2201
6.459 <code>std::function< _Res(_ ArgTypes...)></code> Class Template Reference	2202
6.459.1 Detailed Description	2203
6.459.2 Constructor & Destructor Documentation	2203
6.459.3 Member Function Documentation	2205
6.460 <code>std::function_ref< _Res(_ ArgTypes...) _GLIBCXX_MOF_CV noexcept(_ Noex)></code> Class Template Reference	2208
6.460.1 Detailed Description	2209
6.460.2 Constructor & Destructor Documentation	2209
6.460.3 Member Function Documentation	2210
6.461 <code>std::future< _Res ></code> Class Template Reference	2210
6.461.1 Detailed Description	2212
6.461.2 Constructor & Destructor Documentation	2212
6.461.3 Member Function Documentation	2212
6.462 <code>std::future< _Res & ></code> Class Template Reference	2212
6.462.1 Detailed Description	2213
6.462.2 Constructor & Destructor Documentation	2213
6.462.3 Member Function Documentation	2213
6.463 <code>std::future< void ></code> Class Reference	2214
6.463.1 Detailed Description	2215
6.463.2 Constructor & Destructor Documentation	2215
6.463.3 Member Function Documentation	2215
6.464 <code>std::future_error</code> Class Reference	2215
6.464.1 Detailed Description	2216
6.464.2 Member Function Documentation	2216
6.465 <code>std::gamma_distribution< _RealType ></code> Class Template Reference	2216
6.465.1 Detailed Description	2217
6.465.2 Member Typedef Documentation	2218
6.465.3 Constructor & Destructor Documentation	2218
6.465.4 Member Function Documentation	2218
6.465.5 Friends And Related Symbol Documentation	2219
6.466 <code>std::geometric_distribution< _IntType ></code> Class Template Reference	2220
6.466.1 Detailed Description	2221

6.466.2 Member Typedef Documentation	2221
6.466.3 Member Function Documentation	2222
6.466.4 Friends And Related Symbol Documentation	2223
6.467 <code>__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc ></code> Class Template Reference	2223
6.467.1 Detailed Description	2224
6.467.2 Constructor & Destructor Documentation	2224
6.468 <code>__gnu_pbds::gp_hash_tag</code> Struct Reference	2228
6.468.1 Detailed Description	2229
6.469 <code>__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy ></code> Class Template Reference	2229
6.469.1 Detailed Description	2232
6.469.2 Member Enumeration Documentation	2232
6.469.3 Member Function Documentation	2232
6.470 <code>std::chrono::gps_clock</code> Class Reference	2234
6.470.1 Detailed Description	2234
6.471 <code>std::greater< _Tp ></code> Struct Template Reference	2235
6.471.1 Detailed Description	2235
6.471.2 Member Typedef Documentation	2235
6.472 <code>std::ranges::greater</code> Struct Reference	2236
6.472.1 Detailed Description	2236
6.473 <code>std::greater< void ></code> Struct Reference	2236
6.473.1 Detailed Description	2237
6.473.2 Member Typedef Documentation	2237
6.474 <code>std::greater_equal< _Tp ></code> Struct Template Reference	2237
6.474.1 Detailed Description	2238
6.474.2 Member Typedef Documentation	2238
6.475 <code>std::ranges::greater_equal</code> Struct Reference	2238
6.475.1 Detailed Description	2238
6.476 <code>std::greater_equal< void ></code> Struct Reference	2238
6.476.1 Detailed Description	2239
6.476.2 Member Typedef Documentation	2239
6.477 <code>__gnu_cxx::random_condition::group_adjustor</code> Struct Reference	2240
6.477.1 Detailed Description	2240
6.478 <code>__gnu_parallel::growing_blocks_tag</code> Struct Reference	2240
6.478.1 Detailed Description	2240
6.479 <code>std::gslice</code> Class Reference	2240
6.479.1 Detailed Description	2241
6.480 <code>std::gslice_array< _Tp ></code> Class Template Reference	2241
6.480.1 Detailed Description	2242

6.480.2 Member Function Documentation	2242
6.481 std::has_virtual_destructor< _Tp > Struct Template Reference	2244
6.481.1 Detailed Description	2244
6.482 std::hash< _Tp > Struct Template Reference	2244
6.482.1 Detailed Description	2244
6.483 std::hash< __debug::bitset< _Nb > > Struct Template Reference	2244
6.483.1 Detailed Description	2244
6.484 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference	2245
6.484.1 Detailed Description	2245
6.485 std::hash< __gnu_cxx::__u16vstring > Struct Reference	2245
6.485.1 Detailed Description	2245
6.486 std::hash< __gnu_cxx::__u32vstring > Struct Reference	2245
6.486.1 Detailed Description	2245
6.487 std::hash< __gnu_cxx::__vstring > Struct Reference	2245
6.487.1 Detailed Description	2246
6.488 std::hash< __gnu_cxx::__wvstring > Struct Reference	2246
6.488.1 Detailed Description	2246
6.489 std::hash< __gnu_cxx::throw_value_limit > Struct Reference	2246
6.489.1 Detailed Description	2247
6.489.2 Member Typedef Documentation	2247
6.490 std::hash< __gnu_cxx::throw_value_random > Struct Reference	2247
6.490.1 Detailed Description	2247
6.490.2 Member Typedef Documentation	2248
6.491 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference	2248
6.491.1 Detailed Description	2248
6.492 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference	2248
6.492.1 Detailed Description	2249
6.493 std::hash< _Tp * > Struct Template Reference	2249
6.493.1 Detailed Description	2249
6.494 std::hash< basic_string< char, char_traits< char >, _Alloc > > Struct Template Reference	2249
6.494.1 Detailed Description	2249
6.495 std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > > Struct Template Reference	2249
6.495.1 Detailed Description	2250
6.496 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference	2250
6.496.1 Detailed Description	2250
6.497 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference	2250
6.497.1 Detailed Description	2250
6.498 std::hash< bool > Struct Reference	2250
6.498.1 Detailed Description	2251

6.499	std::hash< char > Struct Reference	2251
6.499.1	Detailed Description	2251
6.500	std::hash< char16_t > Struct Reference	2251
6.500.1	Detailed Description	2251
6.501	std::hash< char32_t > Struct Reference	2251
6.501.1	Detailed Description	2251
6.502	std::hash< double > Struct Reference	2251
6.502.1	Detailed Description	2252
6.503	std::hash< error_code > Struct Reference	2252
6.503.1	Detailed Description	2252
6.504	std::hash< error_condition > Struct Reference	2252
6.504.1	Detailed Description	2252
6.505	std::hash< experimental::optional< _Tp > > Struct Template Reference	2252
6.505.1	Detailed Description	2253
6.506	std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference	2253
6.506.1	Detailed Description	2253
6.507	std::hash< float > Struct Reference	2253
6.507.1	Detailed Description	2253
6.508	std::hash< int > Struct Reference	2253
6.508.1	Detailed Description	2253
6.509	std::hash< long > Struct Reference	2254
6.509.1	Detailed Description	2254
6.510	std::hash< long double > Struct Reference	2254
6.510.1	Detailed Description	2254
6.511	std::hash< long long > Struct Reference	2254
6.511.1	Detailed Description	2254
6.512	std::hash< shared_ptr< _Tp > > Struct Template Reference	2254
6.512.1	Detailed Description	2255
6.513	std::hash< short > Struct Reference	2255
6.513.1	Detailed Description	2255
6.514	std::hash< signed char > Struct Reference	2255
6.514.1	Detailed Description	2255
6.515	std::hash< thread::id > Struct Reference	2255
6.515.1	Detailed Description	2255
6.516	std::hash< type_index > Struct Reference	2255
6.516.1	Detailed Description	2256
6.517	std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	2256
6.517.1	Detailed Description	2256
6.518	std::hash< unsigned char > Struct Reference	2256

6.518.1 Detailed Description	2256
6.519 std::hash< unsigned int > Struct Reference	2256
6.519.1 Detailed Description	2256
6.520 std::hash< unsigned long > Struct Reference	2257
6.520.1 Detailed Description	2257
6.521 std::hash< unsigned long long > Struct Reference	2257
6.521.1 Detailed Description	2257
6.522 std::hash< unsigned short > Struct Reference	2257
6.522.1 Detailed Description	2257
6.523 std::hash< wchar_t > Struct Reference	2257
6.523.1 Detailed Description	2257
6.524 std::hash<::bitset< _Nb > > Struct Template Reference	2258
6.524.1 Detailed Description	2258
6.525 std::hash<::vector< bool, _Alloc > > Struct Template Reference	2258
6.525.1 Detailed Description	2258
6.526 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference	2259
6.526.1 Detailed Description	2259
6.527 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference	2259
6.527.1 Detailed Description	2260
6.528 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference	2260
6.528.1 Detailed Description	2260
6.529 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference	2260
6.529.1 Detailed Description	2261
6.529.2 Constructor & Destructor Documentation	2261
6.530 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference	2261
6.530.1 Detailed Description	2262
6.530.2 Member Enumeration Documentation	2263
6.530.3 Constructor & Destructor Documentation	2263
6.530.4 Member Function Documentation	2263
6.531 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference	2264
6.531.1 Detailed Description	2264
6.532 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference	2264
6.532.1 Detailed Description	2265
6.533 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2265
6.533.1 Detailed Description	2266
6.534 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2267
6.534.1 Detailed Description	2268

6.535 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference	2268
6.535.1 Detailed Description	2269
6.536 __gnu_pbds::hash_prime_size_policy Class Reference	2270
6.536.1 Detailed Description	2270
6.536.2 Member Typedef Documentation	2270
6.536.3 Constructor & Destructor Documentation	2270
6.537 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference	2270
6.537.1 Detailed Description	2272
6.538 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference	2272
6.538.1 Detailed Description	2273
6.538.2 Member Enumeration Documentation	2273
6.538.3 Constructor & Destructor Documentation	2274
6.538.4 Member Function Documentation	2274
6.539 std::chrono::hh_mm_ss< _Duration > Class Template Reference	2276
6.539.1 Detailed Description	2276
6.540 std::locale::id Class Reference	2276
6.540.1 Detailed Description	2277
6.540.2 Constructor & Destructor Documentation	2277
6.540.3 Friends And Related Symbol Documentation	2277
6.541 std::thread::id Class Reference	2278
6.541.1 Detailed Description	2278
6.542 std::identity Struct Reference	2279
6.542.1 Detailed Description	2279
6.543 std::experimental::fundamentals_v1::in_place_t Struct Reference	2279
6.543.1 Detailed Description	2279
6.544 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference	2279
6.544.1 Detailed Description	2280
6.544.2 Member Typedef Documentation	2280
6.544.3 Constructor & Destructor Documentation	2280
6.544.4 Member Function Documentation	2281
6.544.5 Friends And Related Symbol Documentation	2283
6.545 std::indirect_array< _Tp > Class Template Reference	2283
6.545.1 Detailed Description	2284
6.545.2 Member Function Documentation	2285
6.546 std::initializer_list< _E > Class Template Reference	2286
6.546.1 Detailed Description	2286
6.547 std::inout_ptr_t< _Smart, _Pointer, _Args > Class Template Reference	2287
6.547.1 Detailed Description	2287

6.548 <code>std::input_iterator_tag</code> Struct Reference	2287
6.548.1 Detailed Description	2288
6.549 <code>__gnu_pbds::insert_error</code> Struct Reference	2288
6.549.1 Detailed Description	2289
6.549.2 Member Function Documentation	2289
6.550 <code>std::insert_iterator< _Container ></code> Class Template Reference	2289
6.550.1 Detailed Description	2290
6.550.2 Member Typedef Documentation	2291
6.550.3 Constructor & Destructor Documentation	2291
6.550.4 Member Function Documentation	2291
6.551 <code>std::integer_sequence< _Tp, _Idx ></code> Struct Template Reference	2292
6.551.1 Detailed Description	2292
6.552 <code>std::integral_constant< _Tp, __v ></code> Struct Template Reference	2293
6.552.1 Detailed Description	2293
6.553 <code>std::invalid_argument</code> Class Reference	2293
6.553.1 Detailed Description	2294
6.553.2 Member Function Documentation	2294
6.554 <code>std::ios_base</code> Class Reference	2294
6.554.1 Detailed Description	2297
6.554.2 Member Typedef Documentation	2298
6.554.3 Member Enumeration Documentation	2299
6.554.4 Constructor & Destructor Documentation	2300
6.554.5 Member Function Documentation	2300
6.554.6 Member Data Documentation	2305
6.555 <code>std::is_abstract< _Tp ></code> Struct Template Reference	2309
6.555.1 Detailed Description	2310
6.556 <code>std::is_arithmetic< _Tp ></code> Struct Template Reference	2310
6.556.1 Detailed Description	2310
6.557 <code>std::is_array< _Tp ></code> Struct Template Reference	2310
6.557.1 Detailed Description	2310
6.558 <code>std::is_assignable< _Tp, _Up ></code> Struct Template Reference	2310
6.558.1 Detailed Description	2310
6.559 <code>std::is_base_of< _Base, _Derived ></code> Struct Template Reference	2310
6.559.1 Detailed Description	2311
6.560 <code>std::is_bind_expression< _Tp ></code> Struct Template Reference	2311
6.560.1 Detailed Description	2311
6.561 <code>std::is_bind_expression< _Bind< _Signature > ></code> Struct Template Reference	2311
6.561.1 Detailed Description	2311
6.562 <code>std::is_bind_expression< _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2311

6.562.1 Detailed Description	2311
6.563 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference	2312
6.563.1 Detailed Description	2312
6.564 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference	2312
6.564.1 Detailed Description	2312
6.565 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference	2312
6.565.1 Detailed Description	2312
6.566 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference	2312
6.566.1 Detailed Description	2312
6.567 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference	2313
6.567.1 Detailed Description	2313
6.568 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference	2313
6.568.1 Detailed Description	2313
6.569 std::is_class< _Tp > Struct Template Reference	2313
6.569.1 Detailed Description	2313
6.570 std::is_compound< _Tp > Struct Template Reference	2313
6.570.1 Detailed Description	2313
6.571 std::is_const< _Tp > Struct Template Reference	2314
6.571.1 Detailed Description	2314
6.572 std::is_constructible< _Tp, _Args > Struct Template Reference	2314
6.572.1 Detailed Description	2314
6.573 std::is_copy_assignable< _Tp > Struct Template Reference	2314
6.573.1 Detailed Description	2314
6.574 std::is_copy_constructible< _Tp > Struct Template Reference	2314
6.574.1 Detailed Description	2314
6.575 std::is_default_constructible< _Tp > Struct Template Reference	2314
6.575.1 Detailed Description	2315
6.576 std::is_destructible< _Tp > Struct Template Reference	2315
6.576.1 Detailed Description	2315
6.577 std::is_empty< _Tp > Struct Template Reference	2315
6.577.1 Detailed Description	2315
6.578 std::is_enum< _Tp > Struct Template Reference	2315
6.578.1 Detailed Description	2315
6.579 std::is_error_code_enum< _Tp > Struct Template Reference	2315
6.579.1 Detailed Description	2316
6.580 std::is_error_code_enum< future_errc > Struct Reference	2316
6.580.1 Detailed Description	2316
6.581 std::is_error_condition_enum< _Tp > Struct Template Reference	2316

6.581.1 Detailed Description	2316
6.582 std::is_floating_point< _Tp > Struct Template Reference	2316
6.582.1 Detailed Description	2317
6.583 std::is_function< _Tp > Struct Template Reference	2317
6.583.1 Detailed Description	2317
6.584 std::is_fundamental< _Tp > Struct Template Reference	2317
6.584.1 Detailed Description	2317
6.585 std::is_integral< _Tp > Struct Template Reference	2317
6.585.1 Detailed Description	2317
6.586 std::is_layout_compatible< _Tp, _Up > Struct Template Reference	2317
6.586.1 Detailed Description	2318
6.587 std::is_literal_type< _Tp > Struct Template Reference	2318
6.587.1 Detailed Description	2318
6.588 std::is_lvalue_reference< typename > Struct Template Reference	2318
6.588.1 Detailed Description	2318
6.589 std::is_member_function_pointer< _Tp > Struct Template Reference	2318
6.589.1 Detailed Description	2318
6.590 std::is_member_object_pointer< _Tp > Struct Template Reference	2319
6.590.1 Detailed Description	2319
6.591 std::is_member_pointer< _Tp > Struct Template Reference	2319
6.591.1 Detailed Description	2319
6.592 std::is_move_assignable< _Tp > Struct Template Reference	2319
6.592.1 Detailed Description	2319
6.593 std::is_move_constructible< _Tp > Struct Template Reference	2319
6.593.1 Detailed Description	2319
6.594 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference	2319
6.594.1 Detailed Description	2320
6.595 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference	2320
6.595.1 Detailed Description	2320
6.596 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference	2320
6.596.1 Detailed Description	2320
6.597 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference	2320
6.597.1 Detailed Description	2320
6.598 std::is_nothrow_default_constructible< _Tp > Struct Template Reference	2320
6.598.1 Detailed Description	2321
6.599 std::is_nothrow_destructible< _Tp > Struct Template Reference	2321
6.599.1 Detailed Description	2321
6.600 std::is_nothrow_move_assignable< _Tp > Struct Template Reference	2321
6.600.1 Detailed Description	2321

6.601 std::is_nothrow_move_constructible< _Tp > Struct Template Reference	2321
6.601.1 Detailed Description	2321
6.602 std::is_object< _Tp > Struct Template Reference	2321
6.602.1 Detailed Description	2322
6.603 std::is_placeholder< _Tp > Struct Template Reference	2322
6.603.1 Detailed Description	2322
6.604 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference	2323
6.604.1 Detailed Description	2323
6.605 std::is_pod< _Tp > Struct Template Reference	2324
6.605.1 Detailed Description	2324
6.606 std::is_pointer< _Tp > Struct Template Reference	2324
6.606.1 Detailed Description	2324
6.607 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference	2324
6.607.1 Detailed Description	2324
6.608 std::is_polymorphic< _Tp > Struct Template Reference	2324
6.608.1 Detailed Description	2325
6.609 std::is_reference< _Tp > Struct Template Reference	2325
6.609.1 Detailed Description	2325
6.610 std::is_rvalue_reference< typename > Struct Template Reference	2325
6.610.1 Detailed Description	2325
6.611 std::is_same< _Tp, _Up > Struct Template Reference	2325
6.611.1 Detailed Description	2325
6.612 std::is_scalar< _Tp > Struct Template Reference	2325
6.612.1 Detailed Description	2326
6.613 std::is_signed< _Tp > Struct Template Reference	2326
6.613.1 Detailed Description	2326
6.614 std::is_standard_layout< _Tp > Struct Template Reference	2326
6.614.1 Detailed Description	2326
6.615 std::is_trivial< _Tp > Struct Template Reference	2326
6.615.1 Detailed Description	2326
6.616 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference	2326
6.616.1 Detailed Description	2327
6.617 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference	2327
6.617.1 Detailed Description	2327
6.618 std::is_trivially_copy_assignable< _Tp > Struct Template Reference	2327
6.618.1 Detailed Description	2327
6.619 std::is_trivially_copy_constructible< _Tp > Struct Template Reference	2327
6.619.1 Detailed Description	2327
6.620 std::is_trivially_copyable< _Tp > Struct Template Reference	2327

6.620.1 Detailed Description	2328
6.621 std::is_trivially_default_constructible<_Tp> Struct Template Reference	2328
6.621.1 Detailed Description	2328
6.622 std::is_trivially_destructible<_Tp> Struct Template Reference	2328
6.622.1 Detailed Description	2328
6.623 std::is_trivially_move_assignable<_Tp> Struct Template Reference	2328
6.623.1 Detailed Description	2328
6.624 std::is_trivially_move_constructible<_Tp> Struct Template Reference	2328
6.624.1 Detailed Description	2329
6.625 std::is_union<_Tp> Struct Template Reference	2329
6.625.1 Detailed Description	2329
6.626 std::is_unsigned<_Tp> Struct Template Reference	2329
6.626.1 Detailed Description	2329
6.627 std::is_void<_Tp> Struct Template Reference	2329
6.627.1 Detailed Description	2329
6.628 std::is_volatile<_Tp> Struct Template Reference	2329
6.628.1 Detailed Description	2330
6.629 std::istream_iterator<_Tp, _CharT, _Traits, _Dist> Class Template Reference	2330
6.629.1 Detailed Description	2331
6.629.2 Member Typedef Documentation	2331
6.629.3 Constructor & Destructor Documentation	2331
6.629.4 Friends And Related Symbol Documentation	2332
6.630 std::istreambuf_iterator<_CharT, _Traits> Class Template Reference	2332
6.630.1 Detailed Description	2333
6.630.2 Member Typedef Documentation	2334
6.630.3 Constructor & Destructor Documentation	2335
6.630.4 Member Function Documentation	2335
6.631 __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator Struct Reference	2336
6.631.1 Detailed Description	2337
6.632 std::experimental::filesystem::v1::path::iterator Class Reference	2337
6.632.1 Detailed Description	2337
6.633 std::filesystem::path::iterator Class Reference	2338
6.633.1 Detailed Description	2338
6.634 std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference> Struct Template Reference	2338
6.634.1 Detailed Description	2339
6.634.2 Member Typedef Documentation	2339
6.635 std::iterator_traits<_Iterator> Struct Template Reference	2340
6.635.1 Detailed Description	2340
6.636 std::iterator_traits<_Tp*> Struct Template Reference	2340

6.636.1 Detailed Description	2341
6.637 <code>__gnu_pbds::join_error</code> Struct Reference	2341
6.637.1 Detailed Description	2341
6.637.2 Member Function Documentation	2342
6.638 <code>__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc ></code> Class Template Reference	2342
6.638.1 Detailed Description	2343
6.639 <code>__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc ></code> Class Template Reference	2344
6.639.1 Detailed Description	2344
6.639.2 Member Typedef Documentation	2345
6.639.3 Constructor & Destructor Documentation	2345
6.639.4 Member Function Documentation	2346
6.640 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node< _Value, _Metadata, _Alloc ></code> Struct Template Reference	2347
6.640.1 Detailed Description	2347
6.641 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc ></code> Class Template Reference	2347
6.641.1 Detailed Description	2349
6.641.2 Member Typedef Documentation	2349
6.641.3 Constructor & Destructor Documentation	2350
6.641.4 Member Function Documentation	2350
6.642 <code>std::length_error</code> Class Reference	2351
6.642.1 Detailed Description	2351
6.642.2 Member Function Documentation	2351
6.643 <code>std::less< _Tp ></code> Struct Template Reference	2352
6.643.1 Detailed Description	2352
6.643.2 Member Typedef Documentation	2352
6.644 <code>std::ranges::less</code> Struct Reference	2353
6.644.1 Detailed Description	2353
6.645 <code>std::less< void ></code> Struct Reference	2353
6.645.1 Detailed Description	2354
6.645.2 Member Typedef Documentation	2354
6.646 <code>std::less_equal< _Tp ></code> Struct Template Reference	2354
6.646.1 Detailed Description	2355
6.646.2 Member Typedef Documentation	2355
6.647 <code>std::ranges::less_equal</code> Struct Reference	2355
6.647.1 Detailed Description	2355
6.648 <code>std::less_equal< void ></code> Struct Reference	2355
6.648.1 Detailed Description	2356

6.648.2 Member Typedef Documentation	2356
6.649 <code>__gnu_cxx::limit_condition::limit_adjutor</code> Struct Reference	2357
6.649.1 Detailed Description	2357
6.650 <code>__gnu_cxx::limit_condition</code> Struct Reference	2357
6.650.1 Detailed Description	2357
6.651 <code>std::linear_congruential_engine<_UIntType, __a, __c, __m></code> Class Template Reference	2357
6.651.1 Detailed Description	2358
6.651.2 Member Typedef Documentation	2359
6.651.3 Constructor & Destructor Documentation	2359
6.651.4 Member Function Documentation	2359
6.651.5 Friends And Related Symbol Documentation	2361
6.651.6 Member Data Documentation	2362
6.652 <code>__gnu_pbds::linear_probe_fn<Size_Type></code> Class Template Reference	2362
6.652.1 Detailed Description	2363
6.652.2 Member Function Documentation	2363
6.653 <code>std::__debug::list<_Tp, _Allocator></code> Class Template Reference	2363
6.653.1 Detailed Description	2365
6.654 <code>std::list<_Tp, _Alloc></code> Class Template Reference	2365
6.654.1 Detailed Description	2369
6.654.2 Constructor & Destructor Documentation	2370
6.654.3 Member Function Documentation	2372
6.655 <code>__gnu_pbds::list_update<Key, Mapped, Eq_Fn, Update_Policy, _Alloc></code> Class Template Reference	2387
6.655.1 Detailed Description	2387
6.655.2 Constructor & Destructor Documentation	2388
6.656 <code>__gnu_pbds::list_update_tag</code> Struct Reference	2388
6.656.1 Detailed Description	2388
6.657 <code>std::locale</code> Class Reference	2388
6.657.1 Detailed Description	2390
6.657.2 Member Typedef Documentation	2390
6.657.3 Constructor & Destructor Documentation	2390
6.657.4 Member Function Documentation	2393
6.657.5 Friends And Related Symbol Documentation	2395
6.657.6 Member Data Documentation	2396
6.658 <code>std::lock_guard<_Mutex></code> Class Template Reference	2397
6.658.1 Detailed Description	2398
6.659 <code>std::logic_error</code> Class Reference	2398
6.659.1 Detailed Description	2399
6.659.2 Constructor & Destructor Documentation	2399
6.659.3 Member Function Documentation	2399

6.660 std::logical_and< _Tp > Struct Template Reference	2399
6.660.1 Detailed Description	2400
6.660.2 Member Typedef Documentation	2400
6.661 std::logical_and< void > Struct Reference	2401
6.661.1 Detailed Description	2401
6.661.2 Member Typedef Documentation	2401
6.662 std::logical_not< _Tp > Struct Template Reference	2402
6.662.1 Detailed Description	2402
6.662.2 Member Typedef Documentation	2402
6.663 std::logical_not< void > Struct Reference	2403
6.663.1 Detailed Description	2403
6.663.2 Member Typedef Documentation	2403
6.664 std::logical_or< _Tp > Struct Template Reference	2404
6.664.1 Detailed Description	2404
6.664.2 Member Typedef Documentation	2404
6.665 std::logical_or< void > Struct Reference	2405
6.665.1 Detailed Description	2405
6.665.2 Member Typedef Documentation	2405
6.666 std::lognormal_distribution< _RealType > Class Template Reference	2406
6.666.1 Detailed Description	2407
6.666.2 Member Typedef Documentation	2407
6.666.3 Member Function Documentation	2407
6.666.4 Friends And Related Symbol Documentation	2408
6.667 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference	2409
6.667.1 Detailed Description	2409
6.668 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference	2409
6.668.1 Detailed Description	2410
6.668.2 Member Typedef Documentation	2410
6.668.3 Member Enumeration Documentation	2411
6.668.4 Member Function Documentation	2411
6.669 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference	2411
6.669.1 Detailed Description	2412
6.670 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference	2412
6.670.1 Detailed Description	2415
6.671 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference	2415
6.671.1 Detailed Description	2415
6.671.2 Member Typedef Documentation	2415
6.671.3 Member Function Documentation	2415
6.672 std::make_signed< _Tp > Struct Template Reference	2416

6.672.1 Detailed Description	2416
6.673 std::make_unsigned< _Tp > Struct Template Reference	2416
6.673.1 Detailed Description	2416
6.674 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference	2416
6.674.1 Detailed Description	2417
6.675 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	2417
6.675.1 Detailed Description	2419
6.676 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2419
6.676.1 Detailed Description	2422
6.676.2 Constructor & Destructor Documentation	2423
6.676.3 Member Function Documentation	2426
6.677 std::mask_array< _Tp > Class Template Reference	2441
6.677.1 Detailed Description	2442
6.677.2 Member Function Documentation	2443
6.678 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference	2444
6.678.1 Detailed Description	2445
6.679 std::match_results< _Bi_iter, _Alloc > Class Template Reference	2445
6.679.1 Detailed Description	2447
6.679.2 Constructor & Destructor Documentation	2448
6.679.3 Member Function Documentation	2449
6.680 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	2454
6.680.1 Detailed Description	2455
6.681 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference	2456
6.681.1 Detailed Description	2456
6.682 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2456
6.682.1 Detailed Description	2457
6.682.2 Member Typedef Documentation	2457
6.683 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2458
6.683.1 Detailed Description	2458
6.683.2 Member Typedef Documentation	2458
6.684 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2459
6.684.1 Detailed Description	2459
6.684.2 Member Typedef Documentation	2459
6.685 std::mem_fun_t< _Ret, _Tp > Class Template Reference	2460
6.685.1 Detailed Description	2460
6.685.2 Member Typedef Documentation	2460
6.686 std::pmr::memory_resource Class Reference	2461
6.686.1 Detailed Description	2461

6.687 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference	2461
6.687.1 Detailed Description	2462
6.687.2 Member Typedef Documentation	2464
6.687.3 Constructor & Destructor Documentation	2464
6.687.4 Member Function Documentation	2464
6.687.5 Friends And Related Symbol Documentation	2465
6.688 std::messages< _CharT > Class Template Reference	2466
6.688.1 Detailed Description	2468
6.688.2 Member Typedef Documentation	2468
6.688.3 Constructor & Destructor Documentation	2468
6.688.4 Member Function Documentation	2469
6.688.5 Member Data Documentation	2469
6.689 std::messages_base Struct Reference	2470
6.689.1 Detailed Description	2470
6.690 std::messages_byname< _CharT > Class Template Reference	2470
6.690.1 Detailed Description	2472
6.690.2 Member Function Documentation	2472
6.690.3 Member Data Documentation	2472
6.691 std::minus< _Tp > Struct Template Reference	2472
6.691.1 Detailed Description	2473
6.691.2 Member Typedef Documentation	2473
6.692 std::minus< void > Struct Reference	2474
6.692.1 Detailed Description	2474
6.692.2 Member Typedef Documentation	2474
6.693 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference	2475
6.693.1 Detailed Description	2475
6.694 std::modulus< _Tp > Struct Template Reference	2475
6.694.1 Detailed Description	2476
6.694.2 Member Typedef Documentation	2476
6.695 std::modulus< void > Struct Reference	2476
6.695.1 Detailed Description	2477
6.695.2 Member Typedef Documentation	2477
6.696 std::money_base Class Reference	2477
6.696.1 Detailed Description	2478
6.697 std::money_get< _CharT, _InIter > Class Template Reference	2479
6.697.1 Detailed Description	2480
6.697.2 Member Typedef Documentation	2480
6.697.3 Constructor & Destructor Documentation	2480

6.697.4 Member Function Documentation	2481
6.697.5 Member Data Documentation	2483
6.698 std::money_put< _CharT, _OutIter > Class Template Reference	2483
6.698.1 Detailed Description	2484
6.698.2 Member Typedef Documentation	2484
6.698.3 Constructor & Destructor Documentation	2485
6.698.4 Member Function Documentation	2485
6.698.5 Member Data Documentation	2488
6.699 std::moneypunct< _CharT, _Intl > Class Template Reference	2488
6.699.1 Detailed Description	2490
6.699.2 Member Typedef Documentation	2490
6.699.3 Constructor & Destructor Documentation	2490
6.699.4 Member Function Documentation	2491
6.699.5 Member Data Documentation	2497
6.700 std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2497
6.700.1 Detailed Description	2499
6.700.2 Member Function Documentation	2499
6.700.3 Member Data Documentation	2504
6.701 std::pmr::monotonic_buffer_resource Class Reference	2504
6.701.1 Detailed Description	2505
6.701.2 Member Function Documentation	2506
6.702 std::move_iterator< _Iterator > Class Template Reference	2506
6.702.1 Detailed Description	2507
6.703 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference	2507
6.703.1 Detailed Description	2508
6.703.2 Constructor & Destructor Documentation	2508
6.703.3 Member Function Documentation	2510
6.703.4 Friends And Related Symbol Documentation	2511
6.704 std::move_sentinel< _Sent > Class Template Reference	2511
6.704.1 Detailed Description	2511
6.705 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	2512
6.705.1 Detailed Description	2514
6.706 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2514
6.706.1 Detailed Description	2517
6.706.2 Constructor & Destructor Documentation	2517
6.706.3 Member Function Documentation	2520
6.707 std::multiplies< _Tp > Struct Template Reference	2534
6.707.1 Detailed Description	2535

6.707.2 Member Typedef Documentation	2535
6.708 std::multiplies< void > Struct Reference	2535
6.708.1 Detailed Description	2536
6.708.2 Member Typedef Documentation	2536
6.709 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	2536
6.709.1 Detailed Description	2538
6.710 std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2538
6.710.1 Detailed Description	2541
6.710.2 Constructor & Destructor Documentation	2541
6.710.3 Member Function Documentation	2544
6.711 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	2555
6.711.1 Detailed Description	2556
6.711.2 Member Function Documentation	2556
6.712 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	2556
6.712.1 Detailed Description	2557
6.712.2 Member Function Documentation	2557
6.713 __gnu_parallel::multiway_mergesort_tag Struct Reference	2558
6.713.1 Detailed Description	2558
6.713.2 Member Function Documentation	2558
6.714 std::mutex Class Reference	2559
6.714.1 Detailed Description	2559
6.715 std::negate< _Tp > Struct Template Reference	2559
6.715.1 Detailed Description	2560
6.715.2 Member Typedef Documentation	2560
6.716 std::negate< void > Struct Reference	2560
6.716.1 Detailed Description	2561
6.716.2 Member Typedef Documentation	2561
6.717 std::negative_binomial_distribution< _IntType > Class Template Reference	2561
6.717.1 Detailed Description	2562
6.717.2 Member Typedef Documentation	2562
6.717.3 Member Function Documentation	2562
6.717.4 Friends And Related Symbol Documentation	2563
6.718 std::nested_exception Class Reference	2564
6.718.1 Detailed Description	2565
6.718.2 Constructor & Destructor Documentation	2565
6.718.3 Member Function Documentation	2565
6.719 __gnu_cxx::limit_condition::never_adjustor Struct Reference	2565
6.719.1 Detailed Description	2565
6.720 __gnu_cxx::random_condition::never_adjustor Struct Reference	2565

6.720.1 Detailed Description	2565
6.721 <code>__gnu_cxx::new_allocator< _Tp ></code> Class Template Reference	2566
6.721.1 Detailed Description	2566
6.722 <code>__gnu_pbds::detail::no_throw_copies< Key, Mapped ></code> Struct Template Reference	2567
6.722.1 Detailed Description	2567
6.723 <code>__gnu_pbds::detail::no_throw_copies< Key, null_type ></code> Struct Template Reference	2567
6.723.1 Detailed Description	2567
6.724 <code>std::normal_distribution< _RealType ></code> Class Template Reference	2567
6.724.1 Detailed Description	2568
6.724.2 Member Typedef Documentation	2569
6.724.3 Constructor & Destructor Documentation	2569
6.724.4 Member Function Documentation	2569
6.724.5 Friends And Related Symbol Documentation	2570
6.725 <code>std::not_equal_to< _Tp ></code> Struct Template Reference	2571
6.725.1 Detailed Description	2572
6.725.2 Member Typedef Documentation	2572
6.726 <code>std::ranges::not_equal_to</code> Struct Reference	2573
6.726.1 Detailed Description	2573
6.727 <code>std::not_equal_to< void ></code> Struct Reference	2573
6.727.1 Detailed Description	2574
6.727.2 Member Typedef Documentation	2574
6.728 <code>__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 ></code> Struct Template Reference	2574
6.728.1 Detailed Description	2574
6.729 <code>__gnu_pbds::null_type</code> Struct Reference	2575
6.729.1 Detailed Description	2575
6.730 <code>std::experimental::fundamentals_v1::nullopt_t</code> Struct Reference	2575
6.730.1 Detailed Description	2576
6.731 <code>std::num_get< _CharT, _InIter ></code> Class Template Reference	2576
6.731.1 Detailed Description	2578
6.731.2 Member Typedef Documentation	2578
6.731.3 Constructor & Destructor Documentation	2578
6.731.4 Member Function Documentation	2578
6.731.5 Member Data Documentation	2593
6.732 <code>std::num_put< _CharT, _OutIter ></code> Class Template Reference	2593
6.732.1 Detailed Description	2595
6.732.2 Member Typedef Documentation	2595
6.732.3 Constructor & Destructor Documentation	2595
6.732.4 Member Function Documentation	2596
6.732.5 Member Data Documentation	2606

6.733 std::numeric_limits< _Tp > Struct Template Reference	2606
6.733.1 Detailed Description	2607
6.733.2 Member Function Documentation	2607
6.733.3 Member Data Documentation	2609
6.734 std::numeric_limits< bool > Struct Reference	2611
6.734.1 Detailed Description	2612
6.734.2 Member Data Documentation	2613
6.735 std::numeric_limits< char > Struct Reference	2615
6.735.1 Detailed Description	2616
6.735.2 Member Data Documentation	2617
6.736 std::numeric_limits< char16_t > Struct Reference	2619
6.736.1 Detailed Description	2620
6.736.2 Member Data Documentation	2621
6.737 std::numeric_limits< char32_t > Struct Reference	2623
6.737.1 Detailed Description	2624
6.737.2 Member Data Documentation	2625
6.738 std::numeric_limits< double > Struct Reference	2627
6.738.1 Detailed Description	2628
6.738.2 Member Data Documentation	2629
6.739 std::numeric_limits< float > Struct Reference	2631
6.739.1 Detailed Description	2632
6.739.2 Member Data Documentation	2633
6.740 std::numeric_limits< int > Struct Reference	2635
6.740.1 Detailed Description	2636
6.740.2 Member Data Documentation	2637
6.741 std::numeric_limits< long > Struct Reference	2639
6.741.1 Detailed Description	2640
6.741.2 Member Data Documentation	2641
6.742 std::numeric_limits< long double > Struct Reference	2643
6.742.1 Detailed Description	2644
6.742.2 Member Data Documentation	2645
6.743 std::numeric_limits< long long > Struct Reference	2647
6.743.1 Detailed Description	2648
6.743.2 Member Data Documentation	2649
6.744 std::numeric_limits< short > Struct Reference	2651
6.744.1 Detailed Description	2652
6.744.2 Member Data Documentation	2653
6.745 std::numeric_limits< signed char > Struct Reference	2655
6.745.1 Detailed Description	2656

6.745.2 Member Data Documentation	2657
6.746 std::numeric_limits< unsigned char > Struct Reference	2659
6.746.1 Detailed Description	2660
6.746.2 Member Data Documentation	2661
6.747 std::numeric_limits< unsigned int > Struct Reference	2663
6.747.1 Detailed Description	2664
6.747.2 Member Data Documentation	2665
6.748 std::numeric_limits< unsigned long > Struct Reference	2667
6.748.1 Detailed Description	2668
6.748.2 Member Data Documentation	2669
6.749 std::numeric_limits< unsigned long long > Struct Reference	2671
6.749.1 Detailed Description	2672
6.749.2 Member Data Documentation	2673
6.750 std::numeric_limits< unsigned short > Struct Reference	2675
6.750.1 Detailed Description	2676
6.750.2 Member Data Documentation	2677
6.751 std::numeric_limits< wchar_t > Struct Reference	2679
6.751.1 Detailed Description	2680
6.751.2 Member Data Documentation	2681
6.752 std::num_punct< _CharT > Class Template Reference	2683
6.752.1 Detailed Description	2684
6.752.2 Member Typedef Documentation	2685
6.752.3 Constructor & Destructor Documentation	2685
6.752.4 Member Function Documentation	2686
6.752.5 Member Data Documentation	2688
6.753 std::num_punct_byname< _CharT > Class Template Reference	2689
6.753.1 Detailed Description	2690
6.753.2 Member Function Documentation	2690
6.753.3 Member Data Documentation	2693
6.754 __gnu_parallel::omp_loop_static_tag Struct Reference	2693
6.754.1 Detailed Description	2693
6.754.2 Member Function Documentation	2693
6.755 __gnu_parallel::omp_loop_tag Struct Reference	2694
6.755.1 Detailed Description	2694
6.755.2 Member Function Documentation	2694
6.756 std::once_flag Struct Reference	2695
6.756.1 Detailed Description	2695
6.756.2 Constructor & Destructor Documentation	2695
6.756.3 Member Function Documentation	2695

6.756.4 Friends And Related Symbol Documentation	2695
6.757 std::experimental::fundamentals_v1::optional<_Tp> Class Template Reference	2696
6.757.1 Detailed Description	2697
6.758 std::ostream_iterator<_Tp, _CharT, _Traits> Class Template Reference	2697
6.758.1 Detailed Description	2698
6.758.2 Member Typedef Documentation	2698
6.758.3 Constructor & Destructor Documentation	2699
6.758.4 Member Function Documentation	2700
6.759 std::experimental::fundamentals_v2::ostream_joiner<_DelimT, _CharT, _Traits> Class Template Reference	2700
6.759.1 Detailed Description	2701
6.760 std::ostreambuf_iterator<_CharT, _Traits> Class Template Reference	2701
6.760.1 Detailed Description	2702
6.760.2 Member Typedef Documentation	2702
6.760.3 Constructor & Destructor Documentation	2703
6.760.4 Member Function Documentation	2703
6.761 std::out_of_range Class Reference	2704
6.761.1 Detailed Description	2705
6.761.2 Member Function Documentation	2705
6.762 std::out_ptr_t<_Smart, _Pointer, _Args> Class Template Reference	2705
6.762.1 Detailed Description	2705
6.763 std::output_iterator_tag Struct Reference	2706
6.763.1 Detailed Description	2706
6.764 __gnu_pbds::detail::ov_tree_map<Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc> Class Template Reference	2706
6.764.1 Detailed Description	2708
6.764.2 Member Function Documentation	2708
6.765 __gnu_pbds::detail::ov_tree_node_const_it<Value_Type, Metadata_Type, _Alloc> Class Template Reference	2709
6.765.1 Detailed Description	2710
6.765.2 Member Function Documentation	2710
6.766 __gnu_pbds::detail::ov_tree_node_it<Value_Type, Metadata_Type, _Alloc> Class Template Reference	2711
6.766.1 Detailed Description	2712
6.766.2 Member Function Documentation	2712
6.767 __gnu_pbds::ov_tree_tag Struct Reference	2712
6.767.1 Detailed Description	2713
6.768 std::overflow_error Class Reference	2713
6.768.1 Detailed Description	2714
6.768.2 Member Function Documentation	2714
6.769 std::owner_less<_Tp> Struct Template Reference	2715

6.769.1 Detailed Description	2715
6.770 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference	2715
6.770.1 Detailed Description	2715
6.770.2 Member Typedef Documentation	2715
6.771 std::owner_less< shared_ptr< _Tp > > Struct Template Reference	2716
6.771.1 Detailed Description	2716
6.771.2 Member Typedef Documentation	2716
6.772 std::owner_less< void > Struct Reference	2716
6.772.1 Detailed Description	2717
6.772.2 Member Typedef Documentation	2717
6.773 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference	2717
6.773.1 Detailed Description	2717
6.773.2 Member Typedef Documentation	2717
6.774 std::owner_less< weak_ptr< _Tp > > Struct Template Reference	2718
6.774.1 Detailed Description	2718
6.774.2 Member Typedef Documentation	2718
6.775 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference	2719
6.775.1 Detailed Description	2719
6.776 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference	2719
6.776.1 Detailed Description	2721
6.776.2 Member Typedef Documentation	2721
6.776.3 Constructor & Destructor Documentation	2721
6.776.4 Member Function Documentation	2722
6.776.5 Member Data Documentation	2724
6.777 std::pair< _T1, _T2 > Struct Template Reference	2724
6.777.1 Detailed Description	2726
6.777.2 Member Typedef Documentation	2727
6.777.3 Constructor & Destructor Documentation	2727
6.777.4 Member Function Documentation	2727
6.777.5 Member Data Documentation	2729
6.778 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	2730
6.778.1 Detailed Description	2732
6.779 __gnu_pbds::pairing_heap_tag Struct Reference	2732
6.779.1 Detailed Description	2732
6.780 __gnu_parallel::parallel_tag Struct Reference	2732
6.780.1 Detailed Description	2734
6.780.2 Constructor & Destructor Documentation	2734
6.780.3 Member Function Documentation	2734
6.781 std::bernoulli_distribution::param_type Struct Reference	2734

6.781.1 Detailed Description	2735
6.782 std::binomial_distribution<_IntType>::param_type Struct Reference	2735
6.782.1 Detailed Description	2735
6.783 std::cauchy_distribution<_RealType>::param_type Struct Reference	2735
6.783.1 Detailed Description	2736
6.784 std::chi_squared_distribution<_RealType>::param_type Struct Reference	2736
6.784.1 Detailed Description	2736
6.785 std::discrete_distribution<_IntType>::param_type Struct Reference	2736
6.785.1 Detailed Description	2737
6.786 std::exponential_distribution<_RealType>::param_type Struct Reference	2737
6.786.1 Detailed Description	2737
6.787 std::extreme_value_distribution<_RealType>::param_type Struct Reference	2737
6.787.1 Detailed Description	2738
6.788 std::fisher_f_distribution<_RealType>::param_type Struct Reference	2738
6.788.1 Detailed Description	2738
6.789 std::gamma_distribution<_RealType>::param_type Struct Reference	2738
6.789.1 Detailed Description	2739
6.790 std::geometric_distribution<_IntType>::param_type Struct Reference	2739
6.790.1 Detailed Description	2739
6.791 std::lognormal_distribution<_RealType>::param_type Struct Reference	2739
6.791.1 Detailed Description	2740
6.792 std::negative_binomial_distribution<_IntType>::param_type Struct Reference	2740
6.792.1 Detailed Description	2740
6.793 std::normal_distribution<_RealType>::param_type Struct Reference	2740
6.793.1 Detailed Description	2741
6.794 std::piecewise_constant_distribution<_RealType>::param_type Struct Reference	2741
6.794.1 Detailed Description	2742
6.795 std::piecewise_linear_distribution<_RealType>::param_type Struct Reference	2742
6.795.1 Detailed Description	2742
6.796 std::poisson_distribution<_IntType>::param_type Struct Reference	2742
6.796.1 Detailed Description	2743
6.797 std::student_t_distribution<_RealType>::param_type Struct Reference	2743
6.797.1 Detailed Description	2743
6.798 std::uniform_int_distribution<_IntType>::param_type Struct Reference	2743
6.798.1 Detailed Description	2744
6.799 std::uniform_real_distribution<_RealType>::param_type Struct Reference	2744
6.799.1 Detailed Description	2744
6.800 std::weibull_distribution<_RealType>::param_type Struct Reference	2744
6.800.1 Detailed Description	2745

6.801	__gnu_pbds::detail::pat_trie_base Struct Reference	2745
6.801.1	Detailed Description	2746
6.801.2	Member Enumeration Documentation	2746
6.802	__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference	2746
6.802.1	Detailed Description	2748
6.802.2	Member Enumeration Documentation	2748
6.802.3	Member Function Documentation	2748
6.803	__gnu_pbds::pat_trie_tag Struct Reference	2749
6.803.1	Detailed Description	2750
6.804	std::experimental::filesystem::v1::path Class Reference	2750
6.804.1	Detailed Description	2752
6.805	std::filesystem::path Class Reference	2753
6.805.1	Detailed Description	2755
6.805.2	Member Enumeration Documentation	2755
6.805.3	Friends And Related Symbol Documentation	2756
6.806	std::philox_engine< _UIntType, __w, __n, __r, __consts > Class Template Reference	2756
6.806.1	Detailed Description	2757
6.806.2	Constructor & Destructor Documentation	2758
6.806.3	Member Function Documentation	2758
6.806.4	Friends And Related Symbol Documentation	2759
6.807	std::piecewise_constant_distribution< _RealType > Class Template Reference	2760
6.807.1	Detailed Description	2761
6.807.2	Member Typedef Documentation	2762
6.807.3	Member Function Documentation	2762
6.807.4	Friends And Related Symbol Documentation	2763
6.808	std::piecewise_construct_t Struct Reference	2764
6.808.1	Detailed Description	2764
6.809	std::piecewise_linear_distribution< _RealType > Class Template Reference	2764
6.809.1	Detailed Description	2765
6.809.2	Member Typedef Documentation	2765
6.809.3	Member Function Documentation	2766
6.809.4	Friends And Related Symbol Documentation	2767
6.810	std::plus< _Tp > Struct Template Reference	2768
6.810.1	Detailed Description	2768
6.810.2	Member Typedef Documentation	2768
6.810.3	Member Function Documentation	2769
6.811	__gnu_pbds::point_invalidation_guarantee Struct Reference	2769
6.811.1	Detailed Description	2769

6.812 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	2770
6.812.1 Detailed Description	2770
6.812.2 Member Typedef Documentation	2771
6.813 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference	2771
6.813.1 Detailed Description	2772
6.813.2 Member Typedef Documentation	2772
6.814 std::pointer_traits< _Ptr > Struct Template Reference	2772
6.814.1 Detailed Description	2772
6.814.2 Member Typedef Documentation	2773
6.814.3 Member Function Documentation	2773
6.815 std::pointer_traits< _Tp * > Struct Template Reference	2774
6.815.1 Detailed Description	2774
6.815.2 Member Typedef Documentation	2774
6.815.3 Member Function Documentation	2775
6.816 std::poisson_distribution< _IntType > Class Template Reference	2776
6.816.1 Detailed Description	2777
6.816.2 Member Typedef Documentation	2777
6.816.3 Member Function Documentation	2778
6.816.4 Friends And Related Symbol Documentation	2779
6.817 std::pmr::polymorphic_allocator< _Tp > Class Template Reference	2780
6.817.1 Detailed Description	2780
6.818 std::pmr::pool_options Struct Reference	2781
6.818.1 Detailed Description	2781
6.818.2 Member Data Documentation	2781
6.819 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference	2781
6.819.1 Detailed Description	2782
6.819.2 Constructor & Destructor Documentation	2782
6.820 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	2783
6.820.1 Detailed Description	2784
6.820.2 Constructor & Destructor Documentation	2785
6.820.3 Member Function Documentation	2787
6.821 __gnu_pbds::priority_queue_tag Struct Reference	2788
6.821.1 Detailed Description	2788
6.822 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference	2788
6.822.1 Detailed Description	2788
6.823 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference	2788
6.823.1 Detailed Description	2789
6.823.2 Member Typedef Documentation	2789
6.824 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference	2789

6.824.1 Detailed Description	2789
6.824.2 Member Typedef Documentation	2789
6.825 std::promise< _Res > Class Template Reference	2790
6.825.1 Detailed Description	2790
6.826 std::promise< _Res & > Class Template Reference	2790
6.826.1 Detailed Description	2791
6.827 std::promise< void > Class Reference	2791
6.827.1 Detailed Description	2792
6.828 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference	2792
6.828.1 Detailed Description	2793
6.829 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference	2793
6.829.1 Detailed Description	2793
6.829.2 Member Function Documentation	2793
6.830 std::queue< _Tp, _Sequence > Class Template Reference	2793
6.830.1 Detailed Description	2794
6.830.2 Constructor & Destructor Documentation	2795
6.830.3 Member Function Documentation	2795
6.830.4 Member Data Documentation	2796
6.831 __gnu_parallel::quicksort_tag Struct Reference	2796
6.831.1 Detailed Description	2797
6.831.2 Member Function Documentation	2797
6.832 std::random_access_iterator_tag Struct Reference	2798
6.832.1 Detailed Description	2798
6.833 __gnu_cxx::random_condition Struct Reference	2798
6.833.1 Detailed Description	2799
6.834 std::random_device Class Reference	2799
6.834.1 Detailed Description	2800
6.834.2 Member Typedef Documentation	2800
6.835 std::range_error Class Reference	2800
6.835.1 Detailed Description	2800
6.835.2 Member Function Documentation	2801
6.836 __gnu_pbds::range_invalidation_guarantee Struct Reference	2801
6.836.1 Detailed Description	2801
6.837 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference	2802
6.837.1 Detailed Description	2802
6.838 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference	2802
6.838.1 Detailed Description	2803

6.839 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference	2803
6.839.1 Detailed Description	2803
6.840 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference	2803
6.840.1 Detailed Description	2804
6.841 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference	2804
6.841.1 Detailed Description	2804
6.842 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_↵ Hash > Class Template Reference	2805
6.842.1 Detailed Description	2805
6.843 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference	2805
6.843.1 Detailed Description	2806
6.844 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference	2806
6.844.1 Detailed Description	2806
6.845 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference	2807
6.845.1 Detailed Description	2807
6.846 std::rank< _Tp > Struct Template Reference	2807
6.846.1 Detailed Description	2808
6.847 std::ratio< _Num, _Den > Struct Template Reference	2808
6.847.1 Detailed Description	2808
6.848 std::ratio_equal< _R1, _R2 > Struct Template Reference	2808
6.848.1 Detailed Description	2809
6.849 std::ratio_greater< _R1, _R2 > Struct Template Reference	2809
6.849.1 Detailed Description	2810
6.850 std::ratio_greater_equal< _R1, _R2 > Struct Template Reference	2810
6.850.1 Detailed Description	2811
6.851 std::ratio_less< _R1, _R2 > Struct Template Reference	2811
6.851.1 Detailed Description	2811
6.852 std::ratio_less_equal< _R1, _R2 > Struct Template Reference	2812
6.852.1 Detailed Description	2812
6.853 std::ratio_not_equal< _R1, _R2 > Struct Template Reference	2812
6.853.1 Detailed Description	2813
6.854 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	2813
6.854.1 Detailed Description	2814
6.854.2 Member Typedef Documentation	2814
6.855 __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference	2815

6.855.1 Detailed Description	2818
6.856 <code>__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc ></code> Class Template Reference	2818
6.856.1 Detailed Description	2821
6.856.2 Member Function Documentation	2821
6.857 <code>__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc ></code> Struct Template Reference	2822
6.857.1 Detailed Description	2822
6.858 <code>__gnu_pbds::rb_tree_tag</code> Struct Reference	2822
6.858.1 Detailed Description	2823
6.859 <code>__gnu_pbds::detail::rc< _Node, _Alloc ></code> Class Template Reference	2823
6.859.1 Detailed Description	2824
6.860 <code>__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc ></code> Class Template Reference	2824
6.860.1 Detailed Description	2826
6.861 <code>__gnu_pbds::rc_binomial_heap_tag</code> Struct Reference	2826
6.861.1 Detailed Description	2826
6.862 <code>std::simd::rebind< _Tp, _Mp, _Traits ></code> Struct Template Reference	2826
6.862.1 Detailed Description	2827
6.863 <code>std::simd::rebind< _Tp, _Vp, _Traits ></code> Struct Template Reference	2827
6.863.1 Detailed Description	2827
6.864 <code>__gnu_pbds::detail::rebind_traits< _Alloc, T ></code> Struct Template Reference	2827
6.864.1 Detailed Description	2827
6.865 <code>std::filesystem::recursive_directory_iterator</code> Class Reference	2827
6.865.1 Detailed Description	2828
6.866 <code>__gnu_cxx::recursive_init_error</code> Class Reference	2829
6.866.1 Detailed Description	2829
6.866.2 Member Function Documentation	2829
6.867 <code>std::recursive_mutex</code> Class Reference	2829
6.867.1 Detailed Description	2830
6.868 <code>std::recursive_timed_mutex</code> Class Reference	2830
6.868.1 Detailed Description	2830
6.869 <code>std::bitset< _Nb >::reference</code> Class Reference	2831
6.869.1 Detailed Description	2831
6.870 <code>std::tr2::dynamic_bitset< _WordT, _Alloc >::reference</code> Class Reference	2831
6.870.1 Detailed Description	2832
6.871 <code>std::reference_wrapper< _Tp ></code> Class Template Reference	2832
6.871.1 Detailed Description	2833
6.871.2 Friends And Related Symbol Documentation	2833
6.872 <code>std::regex_error</code> Class Reference	2833
6.872.1 Detailed Description	2834

6.872.2 Constructor & Destructor Documentation	2834
6.872.3 Member Function Documentation	2835
6.873 <code>std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits></code> Class Template Reference	2835
6.873.1 Detailed Description	2836
6.873.2 Constructor & Destructor Documentation	2836
6.873.3 Member Function Documentation	2837
6.874 <code>std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits></code> Class Template Reference	2838
6.874.1 Detailed Description	2838
6.874.2 Constructor & Destructor Documentation	2839
6.874.3 Member Function Documentation	2841
6.875 <code>std::regex_traits<_Ch_type></code> Class Template Reference	2842
6.875.1 Detailed Description	2843
6.875.2 Constructor & Destructor Documentation	2843
6.875.3 Member Function Documentation	2843
6.876 <code>std::remove_all_extents<_Tp></code> Struct Template Reference	2848
6.876.1 Detailed Description	2848
6.877 <code>std::remove_const<_Tp></code> Struct Template Reference	2849
6.877.1 Detailed Description	2849
6.878 <code>std::remove_cv<_Tp></code> Struct Template Reference	2849
6.878.1 Detailed Description	2849
6.879 <code>std::remove_extent<_Tp></code> Struct Template Reference	2849
6.879.1 Detailed Description	2849
6.880 <code>std::remove_pointer<_Tp></code> Struct Template Reference	2849
6.880.1 Detailed Description	2850
6.881 <code>std::remove_reference<_Tp></code> Struct Template Reference	2850
6.881.1 Detailed Description	2850
6.882 <code>std::remove_volatile<_Tp></code> Struct Template Reference	2850
6.882.1 Detailed Description	2850
6.883 <code>__gnu_pbds::resize_error</code> Struct Reference	2850
6.883.1 Detailed Description	2851
6.883.2 Member Function Documentation	2851
6.884 <code>__gnu_pbds::detail::resize_policy<_Tp></code> Class Template Reference	2851
6.884.1 Detailed Description	2852
6.885 <code>std::result_of<_Signature></code> Struct Template Reference	2852
6.885.1 Detailed Description	2852
6.886 <code>std::reverse_iterator<_Iterator></code> Class Template Reference	2852
6.886.1 Detailed Description	2854
6.886.2 Constructor & Destructor Documentation	2854
6.886.3 Member Function Documentation	2855

6.887 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference	2857
6.887.1 Detailed Description	2862
6.888 std::runtime_error Class Reference	2862
6.888.1 Detailed Description	2863
6.888.2 Constructor & Destructor Documentation	2863
6.888.3 Member Function Documentation	2863
6.889 __gnu_pbds::sample_probe_fn Class Reference	2863
6.889.1 Detailed Description	2863
6.889.2 Constructor & Destructor Documentation	2864
6.889.3 Member Function Documentation	2864
6.890 __gnu_pbds::sample_range_hashing Class Reference	2864
6.890.1 Detailed Description	2864
6.890.2 Member Typedef Documentation	2865
6.890.3 Constructor & Destructor Documentation	2865
6.890.4 Member Function Documentation	2865
6.891 __gnu_pbds::sample_ranged_hash_fn Class Reference	2865
6.891.1 Detailed Description	2866
6.891.2 Constructor & Destructor Documentation	2866
6.891.3 Member Function Documentation	2866
6.892 __gnu_pbds::sample_ranged_probe_fn Class Reference	2866
6.892.1 Detailed Description	2867
6.893 __gnu_pbds::sample_resize_policy Class Reference	2867
6.893.1 Detailed Description	2868
6.893.2 Member Typedef Documentation	2868
6.893.3 Constructor & Destructor Documentation	2868
6.893.4 Member Function Documentation	2868
6.894 __gnu_pbds::sample_resize_trigger Class Reference	2870
6.894.1 Detailed Description	2870
6.894.2 Member Typedef Documentation	2870
6.894.3 Constructor & Destructor Documentation	2870
6.894.4 Member Function Documentation	2871
6.895 __gnu_pbds::sample_size_policy Class Reference	2872
6.895.1 Detailed Description	2873
6.895.2 Member Typedef Documentation	2873
6.895.3 Constructor & Destructor Documentation	2873
6.895.4 Member Function Documentation	2873
6.896 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference	2874
6.896.1 Detailed Description	2874

6.897	__gnu_pbds::sample_trie_access_traits Struct Reference	2874
6.897.1	Detailed Description	2874
6.897.2	Member Typedef Documentation	2874
6.897.3	Member Function Documentation	2874
6.898	__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	2875
6.898.1	Detailed Description	2875
6.898.2	Constructor & Destructor Documentation	2875
6.898.3	Member Function Documentation	2875
6.899	__gnu_pbds::sample_update_policy Struct Reference	2876
6.899.1	Detailed Description	2876
6.899.2	Member Typedef Documentation	2876
6.899.3	Constructor & Destructor Documentation	2876
6.899.4	Member Function Documentation	2876
6.900	__gnu_parallel::sampling_tag Struct Reference	2877
6.900.1	Detailed Description	2877
6.900.2	Member Function Documentation	2877
6.901	std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference	2878
6.901.1	Detailed Description	2879
6.902	std::seed_seq Class Reference	2879
6.902.1	Detailed Description	2880
6.902.2	Member Typedef Documentation	2880
6.902.3	Constructor & Destructor Documentation	2880
6.903	__gnu_cxx::select1st< _Pair > Struct Template Reference	2880
6.903.1	Detailed Description	2881
6.903.2	Member Typedef Documentation	2881
6.904	__gnu_cxx::select2nd< _Pair > Struct Template Reference	2881
6.904.1	Detailed Description	2881
6.904.2	Member Typedef Documentation	2881
6.905	__gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference	2882
6.905.1	Detailed Description	2882
6.906	__gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference	2882
6.906.1	Detailed Description	2882
6.907	std::basic_istream< _CharT, _Traits >::sentry Class Reference	2882
6.907.1	Detailed Description	2883
6.907.2	Member Typedef Documentation	2883
6.907.3	Constructor & Destructor Documentation	2883
6.907.4	Member Function Documentation	2883
6.908	std::basic_ostream< _CharT, _Traits >::sentry Class Reference	2884

6.908.1 Detailed Description	2884
6.908.2 Constructor & Destructor Documentation	2884
6.908.3 Member Function Documentation	2885
6.909 <code>__gnu_pbds::sequence_tag</code> Struct Reference	2885
6.909.1 Detailed Description	2885
6.910 <code>__gnu_parallel::sequential_tag</code> Struct Reference	2885
6.910.1 Detailed Description	2886
6.911 <code>std::__debug::set<_Key, _Compare, _Allocator></code> Class Template Reference	2886
6.911.1 Detailed Description	2888
6.912 <code>std::set<_Key, _Compare, _Alloc></code> Class Template Reference	2888
6.912.1 Detailed Description	2890
6.912.2 Member Typedef Documentation	2891
6.912.3 Constructor & Destructor Documentation	2893
6.912.4 Member Function Documentation	2896
6.913 <code>std::shared_future<_Res></code> Class Template Reference	2907
6.913.1 Detailed Description	2908
6.913.2 Constructor & Destructor Documentation	2908
6.913.3 Member Function Documentation	2909
6.914 <code>std::shared_future<_Res &></code> Class Template Reference	2909
6.914.1 Detailed Description	2910
6.914.2 Constructor & Destructor Documentation	2910
6.914.3 Member Function Documentation	2910
6.915 <code>std::shared_future<void></code> Class Reference	2911
6.915.1 Detailed Description	2911
6.915.2 Constructor & Destructor Documentation	2912
6.915.3 Member Function Documentation	2912
6.916 <code>std::shared_lock<_Mutex></code> Class Template Reference	2912
6.916.1 Detailed Description	2913
6.917 <code>std::shared_ptr<_Tp></code> Class Template Reference	2913
6.917.1 Detailed Description	2915
6.917.2 Member Typedef Documentation	2916
6.917.3 Constructor & Destructor Documentation	2916
6.917.4 Member Function Documentation	2922
6.918 <code>std::shared_timed_mutex</code> Class Reference	2923
6.918.1 Detailed Description	2923
6.919 <code>std::shuffle_order_engine<_RandomNumberEngine, __k></code> Class Template Reference	2924
6.919.1 Detailed Description	2924
6.919.2 Member Typedef Documentation	2925
6.919.3 Constructor & Destructor Documentation	2925

6.919.4 Member Function Documentation	2926
6.919.5 Friends And Related Symbol Documentation	2927
6.920 <code>std::ranges::single_view< _Tp ></code> Class Template Reference	2928
6.920.1 Detailed Description	2929
6.921 <code>std::slice</code> Class Reference	2930
6.921.1 Detailed Description	2930
6.921.2 Friends And Related Symbol Documentation	2930
6.922 <code>std::slice_array< _Tp ></code> Class Template Reference	2930
6.922.1 Detailed Description	2931
6.922.2 Member Function Documentation	2932
6.923 <code>__gnu_cxx::slist< _Tp, _Alloc ></code> Class Template Reference	2933
6.923.1 Detailed Description	2935
6.924 <code>std::experimental::filesystem::v1::space_info</code> Struct Reference	2935
6.924.1 Detailed Description	2935
6.925 <code>std::filesystem::space_info</code> Struct Reference	2935
6.925.1 Detailed Description	2935
6.926 <code>__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc ></code> Class Template Reference	2936
6.926.1 Detailed Description	2938
6.926.2 Member Function Documentation	2939
6.927 <code>__gnu_pbds::detail::splay_tree_node< Value_Type, Metadata, _Alloc ></code> Struct Template Reference	2939
6.927.1 Detailed Description	2940
6.928 <code>__gnu_pbds::splay_tree_tag</code> Struct Reference	2940
6.928.1 Detailed Description	2941
6.929 <code>std::stack< _Tp, _Sequence ></code> Class Template Reference	2941
6.929.1 Detailed Description	2942
6.929.2 Constructor & Destructor Documentation	2943
6.929.3 Member Function Documentation	2943
6.930 <code>__gnu_cxx::stdio_filebuf< _CharT, _Traits ></code> Class Template Reference	2944
6.930.1 Detailed Description	2947
6.930.2 Constructor & Destructor Documentation	2947
6.930.3 Member Function Documentation	2948
6.930.4 Member Data Documentation	2964
6.931 <code>__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits ></code> Class Template Reference	2967
6.931.1 Detailed Description	2969
6.931.2 Member Function Documentation	2969
6.931.3 Member Data Documentation	2982
6.932 <code>std::chrono::steady_clock</code> Struct Reference	2982
6.932.1 Detailed Description	2983

6.933	__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference	2983
6.933.1	Detailed Description	2984
6.934	__gnu_pbds::detail::stored_data< _Tv, _Th, false > Struct Template Reference	2984
6.934.1	Detailed Description	2984
6.935	__gnu_pbds::detail::stored_hash< _Th > Struct Template Reference	2985
6.935.1	Detailed Description	2985
6.936	__gnu_pbds::detail::stored_value< _Tv > Struct Template Reference	2985
6.936.1	Detailed Description	2986
6.937	__gnu_pbds::string_tag Struct Reference	2986
6.937.1	Detailed Description	2987
6.938	std::student_t_distribution< _RealType > Class Template Reference	2987
6.938.1	Detailed Description	2988
6.938.2	Member Typedef Documentation	2988
6.938.3	Member Function Documentation	2988
6.938.4	Friends And Related Symbol Documentation	2989
6.939	std::sub_match< _Bilter > Class Template Reference	2990
6.939.1	Detailed Description	2991
6.939.2	Member Function Documentation	2992
6.940	std::ranges::subrange< _It, _Sent, _Kind > Class Template Reference	2994
6.940.1	Detailed Description	2995
6.941	std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference	2995
6.941.1	Detailed Description	2996
6.941.2	Member Typedef Documentation	2996
6.941.3	Constructor & Destructor Documentation	2996
6.941.4	Member Function Documentation	2997
6.941.5	Friends And Related Symbol Documentation	2998
6.942	__gnu_cxx::subtractive_rng Class Reference	2999
6.942.1	Detailed Description	3000
6.942.2	Member Typedef Documentation	3000
6.942.3	Constructor & Destructor Documentation	3000
6.942.4	Member Function Documentation	3000
6.943	__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference	3000
6.943.1	Detailed Description	3001
6.944	std::chrono::system_clock Struct Reference	3001
6.944.1	Detailed Description	3001
6.945	std::system_error Class Reference	3002
6.945.1	Detailed Description	3002
6.945.2	Member Function Documentation	3002
6.946	std::chrono::tai_clock Class Reference	3002

6.946.1 Detailed Description	3003
6.947 __gnu_cxx::temporary_buffer<_ForwardIterator, _Tp> Struct Template Reference	3003
6.947.1 Detailed Description	3004
6.947.2 Constructor & Destructor Documentation	3004
6.947.3 Member Function Documentation	3005
6.948 __gnu_pbds::detail::thin_heap<Value_Type, Cmp_Fn, _Alloc> Class Template Reference	3005
6.948.1 Detailed Description	3007
6.949 __gnu_pbds::thin_heap_tag Struct Reference	3007
6.949.1 Detailed Description	3007
6.950 std::thread Class Reference	3008
6.950.1 Detailed Description	3008
6.950.2 Member Function Documentation	3009
6.951 __gnu_cxx::throw_allocator_base<_Tp, _Cond> Class Template Reference	3009
6.951.1 Detailed Description	3010
6.952 __gnu_cxx::throw_allocator_limit<_Tp> Struct Template Reference	3010
6.952.1 Detailed Description	3011
6.953 __gnu_cxx::throw_allocator_random<_Tp> Struct Template Reference	3012
6.953.1 Detailed Description	3013
6.954 __gnu_cxx::throw_value_base<_Cond> Struct Template Reference	3013
6.954.1 Detailed Description	3014
6.955 __gnu_cxx::throw_value_limit Struct Reference	3014
6.955.1 Detailed Description	3015
6.956 __gnu_cxx::throw_value_random Struct Reference	3015
6.956.1 Detailed Description	3016
6.957 std::time_base Class Reference	3016
6.957.1 Detailed Description	3017
6.958 std::time_get<_CharT, _InIter> Class Template Reference	3017
6.958.1 Detailed Description	3019
6.958.2 Member Typedef Documentation	3019
6.958.3 Constructor & Destructor Documentation	3019
6.958.4 Member Function Documentation	3019
6.958.5 Member Data Documentation	3029
6.959 std::time_get_byname<_CharT, _InIter> Class Template Reference	3029
6.959.1 Detailed Description	3030
6.959.2 Member Function Documentation	3031
6.959.3 Member Data Documentation	3040
6.960 std::chrono::time_point<_Clock, _Dur> Class Template Reference	3040
6.960.1 Detailed Description	3041
6.961 std::time_put<_CharT, _OutIter> Class Template Reference	3041

6.961.1 Detailed Description	3042
6.961.2 Member Typedef Documentation	3042
6.961.3 Constructor & Destructor Documentation	3042
6.961.4 Member Function Documentation	3043
6.961.5 Member Data Documentation	3045
6.962 <code>std::time_put_byname< _CharT, _OutIter ></code> Class Template Reference	3045
6.962.1 Detailed Description	3046
6.962.2 Member Function Documentation	3046
6.962.3 Member Data Documentation	3048
6.963 <code>std::timed_mutex</code> Class Reference	3048
6.963.1 Detailed Description	3049
6.964 <code>std::to_chars_result</code> Struct Reference	3049
6.964.1 Detailed Description	3049
6.965 <code>std::chrono::treat_as_floating_point< _Rep ></code> Struct Template Reference	3049
6.965.1 Detailed Description	3050
6.966 <code>__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc ></code> Class Template Reference	3050
6.966.1 Detailed Description	3051
6.966.2 Member Typedef Documentation	3052
6.966.3 Constructor & Destructor Documentation	3052
6.967 <code>__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp ></code> Struct Template Reference	3053
6.967.1 Detailed Description	3053
6.968 <code>__gnu_pbds::detail::tree_metadata_helper< Node_Update, false ></code> Struct Template Reference	3053
6.968.1 Detailed Description	3053
6.969 <code>__gnu_pbds::detail::tree_metadata_helper< Node_Update, true ></code> Struct Template Reference	3053
6.969.1 Detailed Description	3053
6.970 <code>__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc ></code> Struct Template Reference	3054
6.970.1 Detailed Description	3054
6.971 <code>__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc ></code> Class Template Reference	3054
6.971.1 Detailed Description	3055
6.971.2 Member Function Documentation	3055
6.972 <code>__gnu_pbds::tree_tag</code> Struct Reference	3056
6.972.1 Detailed Description	3056
6.973 <code>__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc ></code> Struct Template Reference	3056
6.973.1 Detailed Description	3056
6.974 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc ></code> Struct Template Reference	3057
6.974.1 Detailed Description	3057

6.974.2 Member Typedef Documentation	3057
6.975 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct</code> Template Reference	3057
6.975.1 Detailed Description	3058
6.975.2 Member Typedef Documentation	3059
6.976 <code>__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct</code> Template Reference	3059
6.976.1 Detailed Description	3060
6.976.2 Member Typedef Documentation	3060
6.977 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct</code> Template Reference	3060
6.977.1 Detailed Description	3060
6.977.2 Member Typedef Documentation	3061
6.978 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct</code> Template Reference	3061
6.978.1 Detailed Description	3062
6.978.2 Member Typedef Documentation	3062
6.979 <code>__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc ></code> Struct Template Reference	3062
6.979.1 Detailed Description	3063
6.979.2 Member Typedef Documentation	3063
6.980 <code>__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference</code>	3063
6.980.1 Detailed Description	3064
6.980.2 Member Typedef Documentation	3065
6.980.3 Constructor & Destructor Documentation	3065
6.981 <code>__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference</code>	3066
6.981.1 Detailed Description	3066
6.982 <code>__gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference</code>	3066
6.982.1 Detailed Description	3066
6.983 <code>__gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference</code>	3066
6.983.1 Detailed Description	3067
6.984 <code>__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc ></code> Struct Template Reference	3067
6.984.1 Detailed Description	3067
6.985 <code>__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Tem-</code> plate Reference	3067
6.985.1 Detailed Description	3069
6.985.2 Member Function Documentation	3069
6.986 <code>__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference</code>	3070
6.986.1 Detailed Description	3071
6.986.2 Member Function Documentation	3071

6.987 <code>__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc ></code> Class Template Reference	3071
6.987.1 Detailed Description	3073
6.987.2 Member Typedef Documentation	3073
6.987.3 Member Function Documentation	3073
6.988 <code>__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc ></code> Struct Template Reference	3074
6.988.1 Detailed Description	3075
6.988.2 Member Typedef Documentation	3075
6.988.3 Member Function Documentation	3075
6.989 <code>__gnu_pbds::trie_tag</code> Struct Reference	3076
6.989.1 Detailed Description	3077
6.990 <code>__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc ></code> Struct Template Reference	3077
6.990.1 Detailed Description	3077
6.991 <code>__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	3078
6.991.1 Detailed Description	3078
6.991.2 Member Typedef Documentation	3078
6.992 <code>__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	3079
6.992.1 Detailed Description	3079
6.992.2 Member Typedef Documentation	3079
6.993 <code>__gnu_pbds::trivial_iterator_tag</code> Struct Reference	3080
6.993.1 Detailed Description	3080
6.994 <code>std::try_to_lock_t</code> Struct Reference	3080
6.994.1 Detailed Description	3080
6.995 <code>std::tuple< _Elements ></code> Class Template Reference	3080
6.995.1 Detailed Description	3082
6.996 <code>std::tuple< _T1, _T2 ></code> Class Template Reference	3082
6.996.1 Detailed Description	3084
6.997 <code>std::tuple_element< __i, _Tp ></code> Struct Template Reference	3084
6.997.1 Detailed Description	3084
6.998 <code>std::tuple_element< 0, pair< _Tp1, _Tp2 > ></code> Struct Template Reference	3084
6.998.1 Detailed Description	3084
6.999 <code>std::tuple_element< 1, pair< _Tp1, _Tp2 > ></code> Struct Template Reference	3084
6.999.1 Detailed Description	3084
6.1000 <code>std::tuple_element< __i, tuple< _Types... > ></code> Struct Template Reference	3085
6.1000.1 Detailed Description	3085
6.1001 <code>std::tuple_element< _Ind, array< _Tp, _Nm > ></code> Struct Template Reference	3085

6.1001.1 Detailed Description	3085
6.1002 std::tuple_size< _Tp > Struct Template Reference	3085
6.1002.1 Detailed Description	3085
6.1003 std::tuple_size< array< _Tp, _Nm > > Struct Template Reference	3085
6.1003.1 Detailed Description	3086
6.1004 std::tuple_size< pair< _Tp1, _Tp2 > > Struct Template Reference	3086
6.1004.1 Detailed Description	3087
6.1005 std::tuple_size< tuple< _Elements... > > Struct Template Reference	3087
6.1005.1 Detailed Description	3088
6.1006 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference	3088
6.1006.1 Detailed Description	3089
6.1007 std::type_index Struct Reference	3089
6.1007.1 Detailed Description	3089
6.1008 __gnu_debug::type_info Class Reference	3089
6.1008.1 Detailed Description	3090
6.1008.2 Constructor & Destructor Documentation	3090
6.1008.3 Member Function Documentation	3090
6.1009 std::type_info Class Reference	3090
6.1009.1 Detailed Description	3091
6.1009.2 Constructor & Destructor Documentation	3091
6.1009.3 Member Function Documentation	3091
6.1010 std::type_order< _Tp, _Up > Struct Template Reference	3091
6.1010.1 Detailed Description	3091
6.1011 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	3092
6.1011.1 Detailed Description	3093
6.1012 std::chrono::tzdb_list Class Reference	3093
6.1012.1 Detailed Description	3093
6.1012.2 Member Function Documentation	3094
6.1013 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference	3094
6.1013.1 Detailed Description	3095
6.1013.2 Member Typedef Documentation	3095
6.1014 std::unary_function< _Arg, _Result > Struct Template Reference	3096
6.1014.1 Detailed Description	3096
6.1014.2 Member Typedef Documentation	3096
6.1015 std::unary_negate< _Predicate > Class Template Reference	3097
6.1015.1 Detailed Description	3097
6.1015.2 Member Typedef Documentation	3097
6.1016 __gnu_parallel::unbalanced_tag Struct Reference	3098
6.1016.1 Detailed Description	3098

6.1016.2 Member Function Documentation	3098
6.1017 std::underflow_error Class Reference	3099
6.1017.1 Detailed Description	3099
6.1017.2 Member Function Documentation	3099
6.1018 std::underlying_type< _Tp > Struct Template Reference	3100
6.1018.1 Detailed Description	3100
6.1019 std::uniform_int_distribution< _IntType > Class Template Reference	3100
6.1019.1 Detailed Description	3101
6.1019.2 Member Typedef Documentation	3101
6.1019.3 Constructor & Destructor Documentation	3101
6.1019.4 Member Function Documentation	3101
6.1019.5 Friends And Related Symbol Documentation	3102
6.1020 std::uniform_real_distribution< _RealType > Class Template Reference	3102
6.1020.1 Detailed Description	3103
6.1020.2 Member Typedef Documentation	3103
6.1020.3 Constructor & Destructor Documentation	3104
6.1020.4 Member Function Documentation	3104
6.1020.5 Friends And Related Symbol Documentation	3105
6.1021 std::unique_lock< _Mutex > Class Template Reference	3105
6.1021.1 Detailed Description	3106
6.1021.2 Friends And Related Symbol Documentation	3106
6.1022 std::unique_ptr< _Tp, _Dp > Class Template Reference	3106
6.1022.1 Detailed Description	3108
6.1022.2 Constructor & Destructor Documentation	3108
6.1022.3 Member Function Documentation	3110
6.1023 std::unique_ptr< _Tp[], _Dp > Class Template Reference	3112
6.1023.1 Detailed Description	3113
6.1023.2 Constructor & Destructor Documentation	3113
6.1023.3 Member Function Documentation	3115
6.1023.4 Friends And Related Symbol Documentation	3117
6.1024 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3118
6.1024.1 Detailed Description	3120
6.1025 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3120
6.1025.1 Detailed Description	3124
6.1025.2 Member Typedef Documentation	3124
6.1025.3 Constructor & Destructor Documentation	3126
6.1025.4 Member Function Documentation	3128
6.1026 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3147
6.1026.1 Detailed Description	3149

6.1027 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3149
6.1027.1 Detailed Description	3153
6.1027.2 Member Typedef Documentation	3153
6.1027.3 Constructor & Destructor Documentation	3155
6.1027.4 Member Function Documentation	3158
6.1028 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . .	3173
6.1028.1 Detailed Description	3175
6.1029 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3175
6.1029.1 Detailed Description	3178
6.1029.2 Member Typedef Documentation	3178
6.1029.3 Constructor & Destructor Documentation	3180
6.1029.4 Member Function Documentation	3183
6.1030 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3196
6.1030.1 Detailed Description	3199
6.1031 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3199
6.1031.1 Detailed Description	3202
6.1031.2 Member Typedef Documentation	3202
6.1031.3 Constructor & Destructor Documentation	3204
6.1031.4 Member Function Documentation	3207
6.1032 std::pmr::unsynchronized_pool_resource Class Reference	3222
6.1032.1 Detailed Description	3222
6.1032.2 Member Function Documentation	3223
6.1033 std::uses_allocator< typename, typename > Struct Template Reference	3223
6.1033.1 Detailed Description	3223
6.1034 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference	3223
6.1034.1 Detailed Description	3223
6.1035 std::chrono::utc_clock Class Reference	3223
6.1035.1 Detailed Description	3224
6.1036 std::valarray< _Tp > Class Template Reference	3224
6.1036.1 Detailed Description	3226
6.1036.2 Constructor & Destructor Documentation	3227
6.1036.3 Member Function Documentation	3227
6.1037 std::__debug::vector< _Tp, _Allocator > Class Template Reference	3230
6.1037.1 Detailed Description	3233
6.1037.2 Constructor & Destructor Documentation	3233
6.1038 std::vector< _Tp, _Alloc > Class Template Reference	3233
6.1038.1 Detailed Description	3237
6.1038.2 Constructor & Destructor Documentation	3237
6.1038.3 Member Function Documentation	3240

6.1039 <code>std::vector< bool, _Alloc ></code> Class Template Reference	3253
6.1039.1 Detailed Description	3256
6.1039.2 Member Function Documentation	3257
6.1040 <code>std::ranges::view_base</code> Struct Reference	3257
6.1040.1 Detailed Description	3257
6.1041 <code>std::ranges::view_interface< _Derived ></code> Class Template Reference	3257
6.1041.1 Detailed Description	3258
6.1042 <code>std::wbuffer_convert< _Codecvt, _Elem, _Tr ></code> Class Template Reference	3258
6.1042.1 Detailed Description	3260
6.1042.2 Member Typedef Documentation	3260
6.1042.3 Constructor & Destructor Documentation	3261
6.1042.4 Member Function Documentation	3262
6.1042.5 Member Data Documentation	3274
6.1043 <code>std::weak_ptr< _Tp ></code> Class Template Reference	3275
6.1043.1 Detailed Description	3276
6.1044 <code>std::weibull_distribution< _RealType ></code> Class Template Reference	3276
6.1044.1 Detailed Description	3277
6.1044.2 Member Typedef Documentation	3278
6.1044.3 Member Function Documentation	3278
6.1044.4 Friends And Related Symbol Documentation	3279
6.1045 <code>std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc ></code> Class Template Reference	3279
6.1045.1 Detailed Description	3280
6.1045.2 Constructor & Destructor Documentation	3280
6.1045.3 Member Function Documentation	3281
7 File Documentation	3283
7.1 <code>algorithm</code> File Reference	3283
7.1.1 Detailed Description	3283
7.2 <code>experimental/algorithm</code> File Reference	3283
7.2.1 Detailed Description	3283
7.3 <code>ext/algorithm</code> File Reference	3283
7.3.1 Detailed Description	3284
7.4 <code>parallel/algorithm</code> File Reference	3285
7.4.1 Detailed Description	3285
7.5 <code>any</code> File Reference	3285
7.5.1 Detailed Description	3285
7.6 <code>experimental/any</code> File Reference	3285
7.6.1 Detailed Description	3286
7.7 <code>array</code> File Reference	3286

7.7.1 Detailed Description	3286
7.8 experimental/array File Reference	3287
7.8.1 Detailed Description	3287
7.9 atomic File Reference	3287
7.9.1 Detailed Description	3291
7.10 auto_ptr.h File Reference	3291
7.10.1 Detailed Description	3291
7.11 backward_warning.h File Reference	3291
7.11.1 Detailed Description	3291
7.12 hash_fun.h File Reference	3291
7.12.1 Detailed Description	3292
7.13 hash_map File Reference	3292
7.13.1 Detailed Description	3292
7.14 hash_set File Reference	3292
7.14.1 Detailed Description	3293
7.15 ostream File Reference	3293
7.15.1 Detailed Description	3293
7.16 barrier File Reference	3293
7.16.1 Detailed Description	3293
7.17 bit File Reference	3294
7.17.1 Detailed Description	3294
7.18 bits/algorithmfwd.h File Reference	3294
7.18.1 Detailed Description	3299
7.19 parallel/algorithmfwd.h File Reference	3299
7.19.1 Detailed Description	3307
7.20 align.h File Reference	3307
7.20.1 Detailed Description	3307
7.21 bits/alloc_traits.h File Reference	3308
7.21.1 Detailed Description	3308
7.22 ext/alloc_traits.h File Reference	3308
7.22.1 Detailed Description	3308
7.23 allocated_ptr.h File Reference	3308
7.23.1 Detailed Description	3308
7.24 allocator.h File Reference	3308
7.24.1 Detailed Description	3308
7.25 atomic_base.h File Reference	3309
7.25.1 Detailed Description	3309
7.26 atomic_futex.h File Reference	3309
7.26.1 Detailed Description	3310

7.27 atomic_lockfree_defines.h File Reference	3310
7.27.1 Detailed Description	3310
7.28 atomic_timed_wait.h File Reference	3310
7.28.1 Detailed Description	3310
7.29 atomic_wait.h File Reference	3310
7.29.1 Detailed Description	3310
7.30 basic_ios.h File Reference	3310
7.30.1 Detailed Description	3310
7.31 basic_ios.tcc File Reference	3311
7.31.1 Detailed Description	3311
7.32 basic_string.h File Reference	3311
7.32.1 Detailed Description	3314
7.33 basic_string.tcc File Reference	3314
7.33.1 Detailed Description	3314
7.34 backward/binders.h File Reference	3314
7.34.1 Detailed Description	3314
7.35 bits/binders.h File Reference	3314
7.35.1 Detailed Description	3315
7.36 boost_concept_check.h File Reference	3315
7.36.1 Detailed Description	3315
7.37 c++0x_warning.h File Reference	3315
7.37.1 Detailed Description	3315
7.38 char_traits.h File Reference	3316
7.38.1 Detailed Description	3316
7.39 charconv.h File Reference	3316
7.39.1 Detailed Description	3316
7.40 chrono.h File Reference	3316
7.40.1 Detailed Description	3320
7.40.2 Function Documentation	3320
7.41 chrono_io.h File Reference	3321
7.41.1 Detailed Description	3324
7.42 codecvt.h File Reference	3324
7.42.1 Detailed Description	3325
7.43 concept_check.h File Reference	3325
7.43.1 Detailed Description	3325
7.44 cow_string.h File Reference	3325
7.44.1 Detailed Description	3325
7.45 cpp_type_traits.h File Reference	3325
7.45.1 Detailed Description	3326

7.46 cpyfunc_impl.h File Reference	3326
7.46.1 Detailed Description	3326
7.47 cxxabi_forced.h File Reference	3326
7.47.1 Detailed Description	3326
7.48 cxxabi_init_exception.h File Reference	3326
7.48.1 Detailed Description	3327
7.49 deque.tcc File Reference	3327
7.49.1 Detailed Description	3328
7.50 enable_special_members.h File Reference	3328
7.50.1 Detailed Description	3329
7.51 erase_if.h File Reference	3329
7.51.1 Detailed Description	3329
7.52 exception.h File Reference	3329
7.52.1 Detailed Description	3329
7.53 exception_defines.h File Reference	3329
7.53.1 Detailed Description	3329
7.54 exception_ptr.h File Reference	3329
7.54.1 Detailed Description	3330
7.54.2 Function Documentation	3330
7.55 formatfwd.h File Reference	3330
7.55.1 Detailed Description	3330
7.56 forward_list.h File Reference	3330
7.56.1 Detailed Description	3331
7.57 forward_list.tcc File Reference	3331
7.57.1 Detailed Description	3331
7.58 bits/fs_dir.h File Reference	3331
7.58.1 Detailed Description	3332
7.59 experimental/bits/fs_dir.h File Reference	3332
7.59.1 Detailed Description	3332
7.60 bits/fs_fwd.h File Reference	3332
7.60.1 Detailed Description	3334
7.61 experimental/bits/fs_fwd.h File Reference	3334
7.61.1 Detailed Description	3336
7.62 bits/fs_ops.h File Reference	3336
7.62.1 Detailed Description	3338
7.63 experimental/bits/fs_ops.h File Reference	3338
7.63.1 Detailed Description	3340
7.64 bits/fs_path.h File Reference	3341
7.64.1 Detailed Description	3341

7.65 experimental/bits/fs_path.h File Reference	3341
7.65.1 Detailed Description	3341
7.66 fstream.tcc File Reference	3341
7.66.1 Detailed Description	3341
7.67 funcref_impl.h File Reference	3342
7.67.1 Detailed Description	3342
7.68 funtexcept.h File Reference	3342
7.68.1 Detailed Description	3342
7.69 functional_hash.h File Reference	3342
7.69.1 Detailed Description	3343
7.70 funcwrap.h File Reference	3343
7.70.1 Detailed Description	3343
7.71 gslice.h File Reference	3344
7.71.1 Detailed Description	3344
7.72 gslice_array.h File Reference	3344
7.72.1 Detailed Description	3344
7.73 hash_bytes.h File Reference	3344
7.73.1 Detailed Description	3344
7.74 backward/hashtable.h File Reference	3344
7.74.1 Detailed Description	3345
7.75 bits/hashtable.h File Reference	3345
7.75.1 Detailed Description	3345
7.76 hashtable_policy.h File Reference	3345
7.76.1 Detailed Description	3345
7.77 indirect.h File Reference	3345
7.77.1 Detailed Description	3345
7.78 indirect_array.h File Reference	3345
7.78.1 Detailed Description	3346
7.79 intcmp.h File Reference	3346
7.79.1 Detailed Description	3346
7.80 invoke.h File Reference	3346
7.80.1 Detailed Description	3347
7.81 ios_base.h File Reference	3347
7.81.1 Detailed Description	3348
7.82 iosfwd_file.h File Reference	3349
7.82.1 Detailed Description	3349
7.83 iosfwd_span.h File Reference	3349
7.83.1 Detailed Description	3349
7.84 iosfwd_string.h File Reference	3349

7.84.1 Detailed Description	3349
7.85 iosfwd_sync.h File Reference	3349
7.85.1 Detailed Description	3349
7.86 istream.tcc File Reference	3350
7.86.1 Detailed Description	3350
7.87 iterator_concepts.h File Reference	3350
7.87.1 Detailed Description	3351
7.88 list.tcc File Reference	3351
7.88.1 Detailed Description	3352
7.89 locale_classes.h File Reference	3352
7.89.1 Detailed Description	3352
7.90 locale_classes.tcc File Reference	3352
7.90.1 Detailed Description	3352
7.91 locale_conv.h File Reference	3353
7.91.1 Detailed Description	3353
7.92 locale_facets.h File Reference	3353
7.92.1 Detailed Description	3354
7.93 locale_facets.tcc File Reference	3354
7.93.1 Detailed Description	3355
7.94 locale_facets_nonio.h File Reference	3355
7.94.1 Detailed Description	3356
7.95 locale_facets_nonio.tcc File Reference	3356
7.95.1 Detailed Description	3356
7.96 localefwd.h File Reference	3356
7.96.1 Detailed Description	3357
7.97 mask_array.h File Reference	3357
7.97.1 Detailed Description	3357
7.98 max_size_type.h File Reference	3357
7.98.1 Detailed Description	3357
7.99 memory_resource.h File Reference	3357
7.99.1 Detailed Description	3358
7.100 memoryfwd.h File Reference	3358
7.100.1 Detailed Description	3358
7.101 mofunc_impl.h File Reference	3358
7.101.1 Detailed Description	3358
7.102 monostate.h File Reference	3358
7.102.1 Detailed Description	3358
7.103 move.h File Reference	3358
7.103.1 Detailed Description	3359

7.104 nested_exception.h File Reference	3359
7.104.1 Detailed Description	3360
7.105 bits/new_allocator.h File Reference	3360
7.105.1 Detailed Description	3360
7.106 ext/new_allocator.h File Reference	3360
7.106.1 Detailed Description	3360
7.107 new_except.h File Reference	3360
7.107.1 Detailed Description	3361
7.108 new_throw.h File Reference	3361
7.108.1 Detailed Description	3361
7.109 node_handle.h File Reference	3361
7.109.1 Detailed Description	3361
7.110 ostream.h File Reference	3361
7.110.1 Detailed Description	3362
7.111 ostream.tcc File Reference	3362
7.111.1 Detailed Description	3363
7.112 ostream_insert.h File Reference	3363
7.112.1 Detailed Description	3363
7.113 ostream_print.h File Reference	3363
7.113.1 Detailed Description	3363
7.114 out_ptr.h File Reference	3363
7.114.1 Detailed Description	3363
7.115 parse_numbers.h File Reference	3364
7.115.1 Detailed Description	3364
7.116 postypes.h File Reference	3364
7.116.1 Detailed Description	3364
7.117 predefined_ops.h File Reference	3364
7.117.1 Detailed Description	3365
7.118 print.h File Reference	3365
7.118.1 Detailed Description	3365
7.119 ptr_traits.h File Reference	3365
7.119.1 Detailed Description	3366
7.120 quoted_string.h File Reference	3366
7.120.1 Detailed Description	3366
7.121 random.h File Reference	3366
7.121.1 Detailed Description	3369
7.122 bits/random.tcc File Reference	3369
7.122.1 Detailed Description	3372
7.123 ext/random.tcc File Reference	3373

7.123.1 Detailed Description	3375
7.124 range_access.h File Reference	3375
7.124.1 Detailed Description	3376
7.125 ranges_algo.h File Reference	3376
7.125.1 Detailed Description	3381
7.126 ranges_algobase.h File Reference	3381
7.126.1 Detailed Description	3382
7.127 ranges_base.h File Reference	3382
7.127.1 Detailed Description	3384
7.127.2 Variable Documentation	3384
7.128 ranges_cmp.h File Reference	3385
7.128.1 Detailed Description	3385
7.129 ranges_uninitialized.h File Reference	3385
7.129.1 Detailed Description	3385
7.130 ranges_util.h File Reference	3385
7.130.1 Detailed Description	3386
7.131 refwrap.h File Reference	3386
7.131.1 Detailed Description	3387
7.132 regex.h File Reference	3387
7.132.1 Detailed Description	3389
7.133 regex.tcc File Reference	3389
7.133.1 Detailed Description	3389
7.134 regex_automaton.h File Reference	3389
7.134.1 Detailed Description	3390
7.135 regex_automaton.tcc File Reference	3390
7.135.1 Detailed Description	3390
7.136 regex_compiler.h File Reference	3390
7.136.1 Detailed Description	3390
7.137 regex_compiler.tcc File Reference	3391
7.137.1 Detailed Description	3391
7.138 regex_constants.h File Reference	3391
7.138.1 Detailed Description	3392
7.139 regex_error.h File Reference	3392
7.139.1 Detailed Description	3393
7.140 regex_executor.h File Reference	3393
7.140.1 Detailed Description	3393
7.141 regex_executor.tcc File Reference	3393
7.141.1 Detailed Description	3394
7.142 regex_scanner.h File Reference	3394

7.142.1 Detailed Description	3394
7.143 regex_scanner.tcc File Reference	3394
7.143.1 Detailed Description	3394
7.144 requires_hosted.h File Reference	3394
7.144.1 Detailed Description	3394
7.145 sat_arith.h File Reference	3394
7.145.1 Detailed Description	3395
7.146 semaphore_base.h File Reference	3395
7.146.1 Detailed Description	3395
7.147 bits/shared_ptr.h File Reference	3395
7.147.1 Detailed Description	3396
7.148 experimental/bits/shared_ptr.h File Reference	3396
7.148.1 Detailed Description	3398
7.149 shared_ptr_atomic.h File Reference	3398
7.149.1 Detailed Description	3399
7.150 shared_ptr_base.h File Reference	3400
7.150.1 Detailed Description	3401
7.151 slice_array.h File Reference	3401
7.151.1 Detailed Description	3401
7.152 specfun.h File Reference	3401
7.152.1 Detailed Description	3403
7.153 sstream.tcc File Reference	3403
7.153.1 Detailed Description	3403
7.154 std_abs.h File Reference	3403
7.154.1 Detailed Description	3404
7.155 std_function.h File Reference	3404
7.155.1 Detailed Description	3404
7.156 std_mutex.h File Reference	3404
7.156.1 Detailed Description	3405
7.157 std_thread.h File Reference	3405
7.157.1 Detailed Description	3405
7.158 stdexcept_except.h File Reference	3405
7.158.1 Detailed Description	3406
7.159 stdexcept_throw.h File Reference	3406
7.159.1 Detailed Description	3406
7.160 stdexcept_throwdef.h File Reference	3406
7.160.1 Detailed Description	3406
7.161 stdexcept_throwfwd.h File Reference	3406
7.161.1 Detailed Description	3406

7.162 stl_algo.h File Reference	3406
7.162.1 Detailed Description	3415
7.163 stl_algo_base.h File Reference	3416
7.163.1 Detailed Description	3421
7.164 stl_bvector.h File Reference	3421
7.164.1 Detailed Description	3421
7.165 stl_construct.h File Reference	3422
7.165.1 Detailed Description	3422
7.166 stl_deque.h File Reference	3422
7.166.1 Detailed Description	3423
7.166.2 Macro Definition Documentation	3423
7.167 stl_function.h File Reference	3423
7.167.1 Detailed Description	3426
7.168 stl_heap.h File Reference	3426
7.168.1 Detailed Description	3427
7.169 bits/stl_iterator.h File Reference	3427
7.169.1 Detailed Description	3430
7.170 debug/stl_iterator.h File Reference	3430
7.170.1 Detailed Description	3431
7.171 stl_iterator_base_funcs.h File Reference	3431
7.171.1 Detailed Description	3431
7.172 stl_iterator_base_types.h File Reference	3432
7.172.1 Detailed Description	3432
7.173 stl_list.h File Reference	3432
7.173.1 Detailed Description	3433
7.174 stl_map.h File Reference	3433
7.174.1 Detailed Description	3434
7.175 stl_multimap.h File Reference	3434
7.175.1 Detailed Description	3435
7.176 stl_multiset.h File Reference	3435
7.176.1 Detailed Description	3436
7.177 stl_numeric.h File Reference	3436
7.177.1 Detailed Description	3436
7.178 stl_pair.h File Reference	3437
7.178.1 Detailed Description	3438
7.179 stl_queue.h File Reference	3438
7.179.1 Detailed Description	3439
7.180 stl_raw_storage_iter.h File Reference	3439
7.180.1 Detailed Description	3439

7.181 std_relops.h File Reference	3439
7.181.1 Detailed Description	3439
7.182 std_set.h File Reference	3440
7.182.1 Detailed Description	3440
7.183 std_stack.h File Reference	3440
7.183.1 Detailed Description	3441
7.184 std_tempbuf.h File Reference	3441
7.184.1 Detailed Description	3442
7.185 std_tree.h File Reference	3442
7.185.1 Detailed Description	3442
7.186 std_uninitialized.h File Reference	3442
7.186.1 Detailed Description	3443
7.187 std_vector.h File Reference	3443
7.187.1 Detailed Description	3444
7.188 stream_iterator.h File Reference	3444
7.188.1 Detailed Description	3444
7.189 streambuf.tcc File Reference	3444
7.189.1 Detailed Description	3444
7.190 streambuf_iterator.h File Reference	3445
7.190.1 Detailed Description	3445
7.191 bits/string_view.tcc File Reference	3445
7.191.1 Detailed Description	3445
7.192 experimental/bits/string_view.tcc File Reference	3446
7.192.1 Detailed Description	3446
7.193 stringfwd.h File Reference	3446
7.193.1 Detailed Description	3446
7.194 text_encoding-data.h File Reference	3446
7.194.1 Detailed Description	3446
7.195 this_thread_sleep.h File Reference	3446
7.195.1 Detailed Description	3447
7.196 unicode-data.h File Reference	3447
7.196.1 Detailed Description	3447
7.197 unicode.h File Reference	3447
7.197.1 Detailed Description	3448
7.198 uniform_int_dist.h File Reference	3449
7.198.1 Detailed Description	3449
7.199 unique_lock.h File Reference	3449
7.199.1 Detailed Description	3449
7.200 unique_ptr.h File Reference	3449

7.200.1 Detailed Description	3450
7.201 unordered_map.h File Reference	3450
7.201.1 Detailed Description	3452
7.202 unordered_set.h File Reference	3453
7.202.1 Detailed Description	3455
7.203 uses_allocator_args.h File Reference	3455
7.203.1 Detailed Description	3455
7.204 utility.h File Reference	3455
7.204.1 Detailed Description	3457
7.205 valarray_after.h File Reference	3457
7.205.1 Detailed Description	3466
7.206 valarray_array.h File Reference	3466
7.206.1 Detailed Description	3474
7.207 valarray_array.tcc File Reference	3474
7.207.1 Detailed Description	3475
7.208 valarray_before.h File Reference	3475
7.208.1 Detailed Description	3475
7.209 vector.tcc File Reference	3475
7.209.1 Detailed Description	3475
7.210 version.h File Reference	3475
7.210.1 Detailed Description	3480
7.211 bitset File Reference	3480
7.211.1 Detailed Description	3481
7.212 debug/bitset File Reference	3481
7.212.1 Detailed Description	3481
7.213 cassert File Reference	3482
7.213.1 Detailed Description	3482
7.214 ccomplex File Reference	3482
7.214.1 Detailed Description	3482
7.215 tr1/ccomplex File Reference	3482
7.215.1 Detailed Description	3482
7.216 cctype File Reference	3482
7.216.1 Detailed Description	3482
7.217 tr1/cctype File Reference	3482
7.217.1 Detailed Description	3482
7.218 cerrno File Reference	3483
7.218.1 Detailed Description	3483
7.219 cenv File Reference	3483
7.219.1 Detailed Description	3483

7.220 tr1/cfenv File Reference	3483
7.220.1 Detailed Description	3483
7.221 cfloat File Reference	3483
7.221.1 Detailed Description	3483
7.222 tr1/cfloat File Reference	3483
7.222.1 Detailed Description	3484
7.223 charconv File Reference	3484
7.223.1 Detailed Description	3485
7.224 chrono File Reference	3485
7.224.1 Detailed Description	3487
7.225 experimental/chrono File Reference	3487
7.225.1 Detailed Description	3488
7.226 cinttypes File Reference	3488
7.226.1 Detailed Description	3488
7.227 tr1/cinttypes File Reference	3488
7.227.1 Detailed Description	3488
7.228 ciso646 File Reference	3488
7.228.1 Detailed Description	3488
7.229 climits File Reference	3488
7.229.1 Detailed Description	3489
7.230 tr1/climits File Reference	3489
7.230.1 Detailed Description	3489
7.231 clocale File Reference	3489
7.231.1 Detailed Description	3489
7.232 cmath File Reference	3489
7.232.1 Detailed Description	3495
7.233 ext/cmath File Reference	3495
7.233.1 Detailed Description	3496
7.234 tr1/cmath File Reference	3496
7.234.1 Detailed Description	3499
7.235 codecvt File Reference	3499
7.235.1 Detailed Description	3499
7.236 complex File Reference	3499
7.236.1 Detailed Description	3503
7.237 tr1/complex File Reference	3503
7.237.1 Detailed Description	3504
7.238 complex.h File Reference	3504
7.238.1 Detailed Description	3504
7.239 Concepts	3504

7.240 condition_variable File Reference	3507
7.240.1 Detailed Description	3507
7.241 csetjmp File Reference	3507
7.241.1 Detailed Description	3507
7.242 csignal File Reference	3507
7.242.1 Detailed Description	3508
7.243 cstdalign File Reference	3508
7.243.1 Detailed Description	3508
7.244 cstdarg File Reference	3508
7.244.1 Detailed Description	3508
7.245 tr1/cstdarg File Reference	3508
7.245.1 Detailed Description	3508
7.246 cstdbool File Reference	3508
7.246.1 Detailed Description	3509
7.247 tr1/cstdbool File Reference	3509
7.247.1 Detailed Description	3509
7.248 cstddef File Reference	3509
7.248.1 Detailed Description	3509
7.249 cstdint File Reference	3509
7.249.1 Detailed Description	3509
7.250 tr1/cstdint File Reference	3509
7.250.1 Detailed Description	3510
7.251 cstdio File Reference	3510
7.251.1 Detailed Description	3510
7.252 tr1/cstdio File Reference	3510
7.252.1 Detailed Description	3510
7.253 cstdlib File Reference	3510
7.253.1 Detailed Description	3510
7.254 tr1/cstdlib File Reference	3511
7.254.1 Detailed Description	3511
7.255 cstring File Reference	3511
7.255.1 Detailed Description	3511
7.256 ctgmath File Reference	3511
7.256.1 Detailed Description	3511
7.257 tr1/ctgmath File Reference	3511
7.257.1 Detailed Description	3511
7.258 ctime File Reference	3512
7.258.1 Detailed Description	3512
7.259 tr1/ctime File Reference	3512

7.259.1 Detailed Description	3512
7.260 cuchar File Reference	3512
7.260.1 Detailed Description	3512
7.261 cwchar File Reference	3512
7.261.1 Detailed Description	3513
7.262 tr1/cwchar File Reference	3513
7.262.1 Detailed Description	3513
7.263 cwctype File Reference	3513
7.263.1 Detailed Description	3513
7.264 tr1/cwctype File Reference	3513
7.264.1 Detailed Description	3513
7.265 assertions.h File Reference	3514
7.265.1 Detailed Description	3514
7.266 debug.h File Reference	3514
7.266.1 Detailed Description	3514
7.267 formatter.h File Reference	3515
7.267.1 Detailed Description	3515
7.268 functions.h File Reference	3515
7.268.1 Detailed Description	3517
7.269 helper_functions.h File Reference	3517
7.269.1 Detailed Description	3518
7.270 macros.h File Reference	3518
7.270.1 Detailed Description	3519
7.270.2 Macro Definition Documentation	3519
7.271 map.h File Reference	3522
7.271.1 Detailed Description	3522
7.272 multimap.h File Reference	3523
7.272.1 Detailed Description	3523
7.273 multiset.h File Reference	3523
7.273.1 Detailed Description	3524
7.274 safe_base.h File Reference	3524
7.274.1 Detailed Description	3524
7.275 safe_container.h File Reference	3524
7.275.1 Detailed Description	3525
7.276 safe_iterator.h File Reference	3525
7.276.1 Detailed Description	3525
7.277 safe_iterator.tcc File Reference	3526
7.277.1 Detailed Description	3527
7.278 safe_local_iterator.h File Reference	3527

7.278.1 Detailed Description	3527
7.279 <code>safe_local_iterator.tcc</code> File Reference	3527
7.279.1 Detailed Description	3527
7.280 <code>safe_sequence.h</code> File Reference	3528
7.280.1 Detailed Description	3528
7.281 <code>safe_sequence.tcc</code> File Reference	3528
7.281.1 Detailed Description	3528
7.282 <code>safe_unordered_base.h</code> File Reference	3528
7.282.1 Detailed Description	3528
7.283 <code>safe_unordered_container.h</code> File Reference	3528
7.283.1 Detailed Description	3528
7.284 <code>safe_unordered_container.tcc</code> File Reference	3529
7.284.1 Detailed Description	3529
7.285 <code>set.h</code> File Reference	3529
7.285.1 Detailed Description	3529
7.286 <code>debugging</code> File Reference	3529
7.286.1 Detailed Description	3529
7.287 <code>decimal</code> File Reference	3530
7.287.1 Detailed Description	3539
7.288 <code>debug/deque</code> File Reference	3539
7.288.1 Detailed Description	3539
7.289 <code>deque</code> File Reference	3539
7.289.1 Detailed Description	3540
7.290 <code>experimental/deque</code> File Reference	3540
7.290.1 Detailed Description	3540
7.291 <code>expected</code> File Reference	3540
7.291.1 Detailed Description	3540
7.292 <code>lfts_config.h</code> File Reference	3540
7.292.1 Detailed Description	3540
7.293 <code>ext/numeric_traits.h</code> File Reference	3540
7.293.1 Detailed Description	3541
7.294 <code>propagate_const</code> File Reference	3541
7.294.1 Detailed Description	3543
7.295 <code>simd</code> File Reference	3543
7.295.1 Detailed Description	3543
7.296 <code>aligned_buffer.h</code> File Reference	3543
7.296.1 Detailed Description	3543
7.297 <code>atomicity.h</code> File Reference	3543
7.297.1 Detailed Description	3543

7.298 bitmap_allocator.h File Reference	3544
7.298.1 Detailed Description	3544
7.298.2 Macro Definition Documentation	3545
7.299 cast.h File Reference	3545
7.299.1 Detailed Description	3545
7.300 codecvt_specializations.h File Reference	3545
7.300.1 Detailed Description	3546
7.301 concurrency.h File Reference	3546
7.301.1 Detailed Description	3546
7.302 debug_allocator.h File Reference	3546
7.302.1 Detailed Description	3546
7.303 enc_filebuf.h File Reference	3547
7.303.1 Detailed Description	3547
7.304 extptr_allocator.h File Reference	3547
7.304.1 Detailed Description	3547
7.305 malloc_allocator.h File Reference	3547
7.305.1 Detailed Description	3547
7.306 mt_allocator.h File Reference	3548
7.306.1 Detailed Description	3548
7.307 assoc_container.hpp File Reference	3548
7.307.1 Detailed Description	3549
7.308 bin_search_tree.hpp File Reference	3549
7.308.1 Detailed Description	3549
7.309 bin_search_tree_/node_iterators.hpp File Reference	3549
7.309.1 Detailed Description	3550
7.310 ov_tree_map_/node_iterators.hpp File Reference	3550
7.310.1 Detailed Description	3550
7.311 point_iterators.hpp File Reference	3550
7.311.1 Detailed Description	3550
7.312 bin_search_tree_/r_erase_fn_imps.hpp File Reference	3550
7.312.1 Detailed Description	3550
7.313 pat_trie_/r_erase_fn_imps.hpp File Reference	3551
7.313.1 Detailed Description	3551
7.314 bin_search_tree_/rotate_fn_imps.hpp File Reference	3551
7.314.1 Detailed Description	3551
7.315 pat_trie_/rotate_fn_imps.hpp File Reference	3551
7.315.1 Detailed Description	3551
7.316 bin_search_tree_/traits.hpp File Reference	3551
7.316.1 Detailed Description	3551

7.317 branch_policy/traits.hpp File Reference	3551
7.317.1 Detailed Description	3551
7.318 ov_tree_map_/traits.hpp File Reference	3551
7.318.1 Detailed Description	3552
7.319 pat_trie_/traits.hpp File Reference	3552
7.319.1 Detailed Description	3552
7.320 rb_tree_map_/traits.hpp File Reference	3552
7.320.1 Detailed Description	3552
7.321 splay_tree_/traits.hpp File Reference	3552
7.321.1 Detailed Description	3552
7.322 binary_heap_/hpp File Reference	3552
7.322.1 Detailed Description	3553
7.323 binary_heap_/const_iterator.hpp File Reference	3553
7.323.1 Detailed Description	3553
7.324 left_child_next_sibling_heap_/const_iterator.hpp File Reference	3553
7.324.1 Detailed Description	3554
7.325 unordered_iterator/const_iterator.hpp File Reference	3554
7.325.1 Detailed Description	3554
7.326 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference	3554
7.326.1 Detailed Description	3554
7.327 binary_heap_/constructors_destructor_fn_imps.hpp File Reference	3554
7.327.1 Detailed Description	3554
7.328 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference	3554
7.328.1 Detailed Description	3554
7.329 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference	3554
7.329.1 Detailed Description	3554
7.330 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference	3554
7.330.1 Detailed Description	3554
7.331 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference	3554
7.331.1 Detailed Description	3554
7.332 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference	3554
7.332.1 Detailed Description	3554
7.333 pat_trie_/constructors_destructor_fn_imps.hpp File Reference	3555
7.333.1 Detailed Description	3555
7.334 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference	3555
7.334.1 Detailed Description	3555
7.335 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference	3555
7.335.1 Detailed Description	3555
7.336 splay_tree_/constructors_destructor_fn_imps.hpp File Reference	3555

7.336.1 Detailed Description	3555
7.337 thin_heap_/constructors_destructor_fn_imps.hpp File Reference	3555
7.337.1 Detailed Description	3555
7.338 bin_search_tree_/debug_fn_imps.hpp File Reference	3555
7.338.1 Detailed Description	3555
7.339 binary_heap_/debug_fn_imps.hpp File Reference	3555
7.339.1 Detailed Description	3555
7.340 binomial_heap_/debug_fn_imps.hpp File Reference	3555
7.340.1 Detailed Description	3555
7.341 binomial_heap_base_/debug_fn_imps.hpp File Reference	3555
7.341.1 Detailed Description	3555
7.342 cc_hash_table_map_/debug_fn_imps.hpp File Reference	3555
7.342.1 Detailed Description	3555
7.343 gp_hash_table_map_/debug_fn_imps.hpp File Reference	3556
7.343.1 Detailed Description	3556
7.344 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference	3556
7.344.1 Detailed Description	3556
7.345 list_update_map_/debug_fn_imps.hpp File Reference	3556
7.345.1 Detailed Description	3556
7.346 ov_tree_map_/debug_fn_imps.hpp File Reference	3556
7.346.1 Detailed Description	3556
7.347 pairing_heap_/debug_fn_imps.hpp File Reference	3556
7.347.1 Detailed Description	3556
7.348 pat_trie_/debug_fn_imps.hpp File Reference	3556
7.348.1 Detailed Description	3556
7.349 rb_tree_map_/debug_fn_imps.hpp File Reference	3556
7.349.1 Detailed Description	3556
7.350 rc_binomial_heap_/debug_fn_imps.hpp File Reference	3556
7.350.1 Detailed Description	3556
7.351 splay_tree_/debug_fn_imps.hpp File Reference	3556
7.351.1 Detailed Description	3556
7.352 thin_heap_/debug_fn_imps.hpp File Reference	3556
7.352.1 Detailed Description	3556
7.353 entry_cmp.hpp File Reference	3557
7.353.1 Detailed Description	3557
7.354 entry_pred.hpp File Reference	3557
7.354.1 Detailed Description	3557
7.355 bin_search_tree_/erase_fn_imps.hpp File Reference	3557
7.355.1 Detailed Description	3557

7.356	binary_heap_/erase_fn_imps.hpp File Reference	3557
7.356.1	Detailed Description	3557
7.357	binomial_heap_base_/erase_fn_imps.hpp File Reference	3557
7.357.1	Detailed Description	3557
7.358	cc_hash_table_map_/erase_fn_imps.hpp File Reference	3557
7.358.1	Detailed Description	3557
7.359	gp_hash_table_map_/erase_fn_imps.hpp File Reference	3558
7.359.1	Detailed Description	3558
7.360	left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference	3558
7.360.1	Detailed Description	3558
7.361	list_update_map_/erase_fn_imps.hpp File Reference	3558
7.361.1	Detailed Description	3558
7.362	ov_tree_map_/erase_fn_imps.hpp File Reference	3558
7.362.1	Detailed Description	3558
7.363	pairing_heap_/erase_fn_imps.hpp File Reference	3558
7.363.1	Detailed Description	3558
7.364	pat_trie_/erase_fn_imps.hpp File Reference	3558
7.364.1	Detailed Description	3558
7.365	rb_tree_map_/erase_fn_imps.hpp File Reference	3558
7.365.1	Detailed Description	3558
7.366	rc_binomial_heap_/erase_fn_imps.hpp File Reference	3558
7.366.1	Detailed Description	3558
7.367	splay_tree_/erase_fn_imps.hpp File Reference	3558
7.367.1	Detailed Description	3558
7.368	thin_heap_/erase_fn_imps.hpp File Reference	3558
7.368.1	Detailed Description	3558
7.369	bin_search_tree_/find_fn_imps.hpp File Reference	3559
7.369.1	Detailed Description	3559
7.370	binary_heap_/find_fn_imps.hpp File Reference	3559
7.370.1	Detailed Description	3559
7.371	binomial_heap_base_/find_fn_imps.hpp File Reference	3559
7.371.1	Detailed Description	3559
7.372	cc_hash_table_map_/find_fn_imps.hpp File Reference	3559
7.372.1	Detailed Description	3559
7.373	gp_hash_table_map_/find_fn_imps.hpp File Reference	3559
7.373.1	Detailed Description	3559
7.374	list_update_map_/find_fn_imps.hpp File Reference	3559
7.374.1	Detailed Description	3559
7.375	pairing_heap_/find_fn_imps.hpp File Reference	3559

7.375.1 Detailed Description	3559
7.376 pat_trie_/find_fn_imps.hpp File Reference	3559
7.376.1 Detailed Description	3559
7.377 rb_tree_map_/find_fn_imps.hpp File Reference	3559
7.377.1 Detailed Description	3559
7.378 splay_tree_/find_fn_imps.hpp File Reference	3559
7.378.1 Detailed Description	3559
7.379 thin_heap_/find_fn_imps.hpp File Reference	3560
7.379.1 Detailed Description	3560
7.380 bin_search_tree_/info_fn_imps.hpp File Reference	3560
7.380.1 Detailed Description	3560
7.381 binary_heap_/info_fn_imps.hpp File Reference	3560
7.381.1 Detailed Description	3560
7.382 cc_hash_table_map_/info_fn_imps.hpp File Reference	3560
7.382.1 Detailed Description	3560
7.383 gp_hash_table_map_/info_fn_imps.hpp File Reference	3560
7.383.1 Detailed Description	3560
7.384 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference	3560
7.384.1 Detailed Description	3560
7.385 list_update_map_/info_fn_imps.hpp File Reference	3560
7.385.1 Detailed Description	3560
7.386 ov_tree_map_/info_fn_imps.hpp File Reference	3560
7.386.1 Detailed Description	3560
7.387 pat_trie_/info_fn_imps.hpp File Reference	3560
7.387.1 Detailed Description	3560
7.388 rb_tree_map_/info_fn_imps.hpp File Reference	3560
7.388.1 Detailed Description	3560
7.389 splay_tree_/info_fn_imps.hpp File Reference	3561
7.389.1 Detailed Description	3561
7.390 bin_search_tree_/insert_fn_imps.hpp File Reference	3561
7.390.1 Detailed Description	3561
7.391 binary_heap_/insert_fn_imps.hpp File Reference	3561
7.391.1 Detailed Description	3561
7.392 binomial_heap_base_/insert_fn_imps.hpp File Reference	3561
7.392.1 Detailed Description	3561
7.393 cc_hash_table_map_/insert_fn_imps.hpp File Reference	3561
7.393.1 Detailed Description	3561
7.394 gp_hash_table_map_/insert_fn_imps.hpp File Reference	3561
7.394.1 Detailed Description	3561

7.395 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference	3561
7.395.1 Detailed Description	3561
7.396 list_update_map_/insert_fn_imps.hpp File Reference	3561
7.396.1 Detailed Description	3561
7.397 ov_tree_map_/insert_fn_imps.hpp File Reference	3561
7.397.1 Detailed Description	3561
7.398 pairing_heap_/insert_fn_imps.hpp File Reference	3561
7.398.1 Detailed Description	3561
7.399 rb_tree_map_/insert_fn_imps.hpp File Reference	3562
7.399.1 Detailed Description	3562
7.400 rc_binomial_heap_/insert_fn_imps.hpp File Reference	3562
7.400.1 Detailed Description	3562
7.401 splay_tree_/insert_fn_imps.hpp File Reference	3562
7.401.1 Detailed Description	3562
7.402 thin_heap_/insert_fn_imps.hpp File Reference	3562
7.402.1 Detailed Description	3562
7.403 bin_search_tree_/iterators_fn_imps.hpp File Reference	3562
7.403.1 Detailed Description	3562
7.404 binary_heap_/iterators_fn_imps.hpp File Reference	3562
7.404.1 Detailed Description	3562
7.405 cc_hash_table_map_/iterators_fn_imps.hpp File Reference	3562
7.405.1 Detailed Description	3562
7.406 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference	3562
7.406.1 Detailed Description	3562
7.407 list_update_map_/iterators_fn_imps.hpp File Reference	3562
7.407.1 Detailed Description	3562
7.408 ov_tree_map_/iterators_fn_imps.hpp File Reference	3562
7.408.1 Detailed Description	3562
7.409 pat_trie_/iterators_fn_imps.hpp File Reference	3563
7.409.1 Detailed Description	3563
7.410 binary_heap_/point_const_iterator.hpp File Reference	3563
7.410.1 Detailed Description	3563
7.411 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference	3563
7.411.1 Detailed Description	3563
7.412 unordered_iterator/point_const_iterator.hpp File Reference	3563
7.412.1 Detailed Description	3563
7.413 bin_search_tree_/policy_access_fn_imps.hpp File Reference	3563
7.413.1 Detailed Description	3563
7.414 binary_heap_/policy_access_fn_imps.hpp File Reference	3563

7.414.1 Detailed Description	3563
7.415 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference	3564
7.415.1 Detailed Description	3564
7.416 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference	3564
7.416.1 Detailed Description	3564
7.417 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference	3564
7.417.1 Detailed Description	3564
7.418 ov_tree_map_/policy_access_fn_imps.hpp File Reference	3564
7.418.1 Detailed Description	3564
7.419 pat_trie_/policy_access_fn_imps.hpp File Reference	3564
7.419.1 Detailed Description	3564
7.420 resize_policy.hpp File Reference	3564
7.420.1 Detailed Description	3564
7.421 bin_search_tree_/split_join_fn_imps.hpp File Reference	3564
7.421.1 Detailed Description	3564
7.422 binary_heap_/split_join_fn_imps.hpp File Reference	3565
7.422.1 Detailed Description	3565
7.423 binomial_heap_base_/split_join_fn_imps.hpp File Reference	3565
7.423.1 Detailed Description	3565
7.424 ov_tree_map_/split_join_fn_imps.hpp File Reference	3565
7.424.1 Detailed Description	3565
7.425 pairing_heap_/split_join_fn_imps.hpp File Reference	3565
7.425.1 Detailed Description	3565
7.426 rb_tree_map_/split_join_fn_imps.hpp File Reference	3565
7.426.1 Detailed Description	3565
7.427 rc_binomial_heap_/split_join_fn_imps.hpp File Reference	3565
7.427.1 Detailed Description	3565
7.428 splay_tree_/split_join_fn_imps.hpp File Reference	3565
7.428.1 Detailed Description	3565
7.429 thin_heap_/split_join_fn_imps.hpp File Reference	3565
7.429.1 Detailed Description	3565
7.430 binary_heap_/trace_fn_imps.hpp File Reference	3565
7.430.1 Detailed Description	3565
7.431 cc_hash_table_map_/trace_fn_imps.hpp File Reference	3565
7.431.1 Detailed Description	3565
7.432 gp_hash_table_map_/trace_fn_imps.hpp File Reference	3566
7.432.1 Detailed Description	3566
7.433 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference	3566
7.433.1 Detailed Description	3566

7.434 list_update_map_/trace_fn_imps.hpp File Reference	3566
7.434.1 Detailed Description	3566
7.435 pat_trie_/trace_fn_imps.hpp File Reference	3566
7.435.1 Detailed Description	3566
7.436 rc_binomial_heap_/trace_fn_imps.hpp File Reference	3566
7.436.1 Detailed Description	3566
7.437 thin_heap_/trace_fn_imps.hpp File Reference	3566
7.437.1 Detailed Description	3566
7.438 binomial_heap_.hpp File Reference	3566
7.438.1 Detailed Description	3566
7.439 binomial_heap_base_.hpp File Reference	3567
7.439.1 Detailed Description	3567
7.440 branch_policy.hpp File Reference	3567
7.440.1 Detailed Description	3567
7.441 null_node_metadata.hpp File Reference	3567
7.441.1 Detailed Description	3567
7.442 cc_ht_map_.hpp File Reference	3567
7.442.1 Detailed Description	3568
7.443 cmp_fn_imps.hpp File Reference	3568
7.443.1 Detailed Description	3568
7.444 cond_key_dtor_entry_dealtor.hpp File Reference	3568
7.444.1 Detailed Description	3569
7.445 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference	3569
7.445.1 Detailed Description	3569
7.446 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference	3569
7.446.1 Detailed Description	3569
7.447 list_update_map_/constructor_destructor_fn_imps.hpp File Reference	3569
7.448 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference	3569
7.448.1 Detailed Description	3569
7.449 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference	3569
7.449.1 Detailed Description	3569
7.450 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference	3569
7.450.1 Detailed Description	3569
7.451 gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference	3569
7.451.1 Detailed Description	3569
7.452 cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference	3569
7.452.1 Detailed Description	3569
7.453 gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference	3569
7.453.1 Detailed Description	3569

7.454 cc_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference	3570
7.454.1 Detailed Description	3570
7.455 gp_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference	3570
7.455.1 Detailed Description	3570
7.456 entry_list_fn_imps.hpp File Reference	3570
7.456.1 Detailed Description	3570
7.457 cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference	3570
7.457.1 Detailed Description	3570
7.458 gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference	3570
7.458.1 Detailed Description	3570
7.459 cc_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference	3570
7.459.1 Detailed Description	3570
7.460 gp_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference	3570
7.460.1 Detailed Description	3570
7.461 cc_hash_table_map_/find_store_hash_fn_imps.hpp File Reference	3570
7.461.1 Detailed Description	3570
7.462 gp_hash_table_map_/find_store_hash_fn_imps.hpp File Reference	3570
7.462.1 Detailed Description	3570
7.463 cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference	3570
7.463.1 Detailed Description	3570
7.464 gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference	3571
7.464.1 Detailed Description	3571
7.465 cc_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference	3571
7.465.1 Detailed Description	3571
7.466 gp_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference	3571
7.466.1 Detailed Description	3571
7.467 cc_hash_table_map_/resize_fn_imps.hpp File Reference	3571
7.467.1 Detailed Description	3571
7.468 gp_hash_table_map_/resize_fn_imps.hpp File Reference	3571
7.468.1 Detailed Description	3571
7.469 cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference	3571
7.469.1 Detailed Description	3571
7.470 gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference	3571
7.470.1 Detailed Description	3571
7.471 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference	3571
7.471.1 Detailed Description	3571
7.472 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference	3571
7.472.1 Detailed Description	3571
7.473 size_fn_imps.hpp File Reference	3571

7.473.1 Detailed Description	3571
7.474 cond_dealtor.hpp File Reference	3572
7.474.1 Detailed Description	3572
7.475 container_base_dispatch.hpp File Reference	3572
7.475.1 Detailed Description	3573
7.476 debug_map_base.hpp File Reference	3573
7.476.1 Detailed Description	3573
7.477 eq_by_less.hpp File Reference	3573
7.477.1 Detailed Description	3573
7.478 hash_eq_fn.hpp File Reference	3573
7.478.1 Detailed Description	3573
7.479 find_no_store_hash_fn_imps.hpp File Reference	3573
7.479.1 Detailed Description	3573
7.480 gp_ht_map_.hpp File Reference	3573
7.480.1 Detailed Description	3574
7.481 iterator_fn_imps.hpp File Reference	3574
7.481.1 Detailed Description	3574
7.482 direct_mask_range_hashing_imp.hpp File Reference	3574
7.482.1 Detailed Description	3574
7.483 direct_mod_range_hashing_imp.hpp File Reference	3574
7.483.1 Detailed Description	3574
7.484 linear_probe_fn_imp.hpp File Reference	3574
7.484.1 Detailed Description	3574
7.485 mask_based_range_hashing.hpp File Reference	3575
7.485.1 Detailed Description	3575
7.486 mod_based_range_hashing.hpp File Reference	3575
7.486.1 Detailed Description	3575
7.487 probe_fn_base.hpp File Reference	3575
7.487.1 Detailed Description	3575
7.488 quadratic_probe_fn_imp.hpp File Reference	3575
7.488.1 Detailed Description	3575
7.489 ranged_hash_fn.hpp File Reference	3576
7.489.1 Detailed Description	3576
7.490 ranged_probe_fn.hpp File Reference	3576
7.490.1 Detailed Description	3576
7.491 sample_probe_fn.hpp File Reference	3576
7.491.1 Detailed Description	3577
7.492 sample_range_hashing.hpp File Reference	3577
7.492.1 Detailed Description	3577

7.493 sample_ranged_hash_fn.hpp File Reference	3577
7.493.1 Detailed Description	3577
7.494 sample_ranged_probe_fn.hpp File Reference	3577
7.494.1 Detailed Description	3577
7.495 left_child_next_sibling_heap_.hpp File Reference	3577
7.495.1 Detailed Description	3578
7.496 left_child_next_sibling_heap_/node.hpp File Reference	3578
7.496.1 Detailed Description	3578
7.497 rb_tree_map_/node.hpp File Reference	3578
7.497.1 Detailed Description	3578
7.498 splay_tree_/node.hpp File Reference	3578
7.498.1 Detailed Description	3579
7.499 entry_metadata_base.hpp File Reference	3579
7.499.1 Detailed Description	3579
7.500 lu_map_.hpp File Reference	3579
7.500.1 Detailed Description	3579
7.501 lu_counter_metadata.hpp File Reference	3580
7.501.1 Detailed Description	3580
7.502 sample_update_policy.hpp File Reference	3580
7.502.1 Detailed Description	3580
7.503 ov_tree_map_.hpp File Reference	3580
7.503.1 Detailed Description	3581
7.504 pairing_heap_.hpp File Reference	3581
7.504.1 Detailed Description	3581
7.505 insert_join_fn_imps.hpp File Reference	3581
7.505.1 Detailed Description	3581
7.506 pat_trie_.hpp File Reference	3581
7.506.1 Detailed Description	3582
7.507 pat_trie_base.hpp File Reference	3582
7.507.1 Detailed Description	3582
7.508 split_fn_imps.hpp File Reference	3583
7.508.1 Detailed Description	3583
7.509 synth_access_traits.hpp File Reference	3583
7.509.1 Detailed Description	3583
7.510 update_fn_imps.hpp File Reference	3583
7.510.1 Detailed Description	3583
7.511 priority_queue_base_dispatch.hpp File Reference	3583
7.511.1 Detailed Description	3584
7.512 rb_tree_.hpp File Reference	3584

7.512.1 Detailed Description	3584
7.513 rc.hpp File Reference	3584
7.513.1 Detailed Description	3584
7.514 rc_binomial_heap_.hpp File Reference	3584
7.514.1 Detailed Description	3585
7.515 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference	3585
7.515.1 Detailed Description	3585
7.516 hash_exponential_size_policy_imp.hpp File Reference	3585
7.516.1 Detailed Description	3585
7.517 hash_load_check_resize_trigger_imp.hpp File Reference	3585
7.517.1 Detailed Description	3585
7.518 hash_load_check_resize_trigger_size_base.hpp File Reference	3585
7.518.1 Detailed Description	3585
7.519 hash_prime_size_policy_imp.hpp File Reference	3585
7.519.1 Detailed Description	3585
7.520 hash_standard_resize_policy_imp.hpp File Reference	3585
7.520.1 Detailed Description	3585
7.521 sample_resize_policy.hpp File Reference	3585
7.521.1 Detailed Description	3586
7.522 sample_resize_trigger.hpp File Reference	3586
7.522.1 Detailed Description	3586
7.523 sample_size_policy.hpp File Reference	3586
7.523.1 Detailed Description	3586
7.524 splay_fn_imps.hpp File Reference	3586
7.524.1 Detailed Description	3586
7.525 splay_tree_.hpp File Reference	3586
7.525.1 Detailed Description	3587
7.526 standard_policies.hpp File Reference	3587
7.526.1 Detailed Description	3587
7.526.2 Enumeration Type Documentation	3587
7.527 thin_heap_.hpp File Reference	3587
7.527.1 Detailed Description	3588
7.528 tree_policy/node_metadata_selector.hpp File Reference	3588
7.528.1 Detailed Description	3588
7.529 trie_policy/node_metadata_selector.hpp File Reference	3588
7.529.1 Detailed Description	3588
7.530 tree_policy/order_statistics_imp.hpp File Reference	3588
7.530.1 Detailed Description	3588
7.531 trie_policy/order_statistics_imp.hpp File Reference	3589

7.531.1 Detailed Description	3589
7.532 sample_tree_node_update.hpp File Reference	3589
7.532.1 Detailed Description	3589
7.533 tree_trace_base.hpp File Reference	3589
7.533.1 Detailed Description	3589
7.534 prefix_search_node_update_imp.hpp File Reference	3589
7.534.1 Detailed Description	3589
7.535 sample_trie_access_traits.hpp File Reference	3589
7.535.1 Detailed Description	3589
7.536 sample_trie_node_update.hpp File Reference	3589
7.536.1 Detailed Description	3590
7.537 trie_policy_base.hpp File Reference	3590
7.537.1 Detailed Description	3590
7.538 trie_string_access_traits_imp.hpp File Reference	3590
7.538.1 Detailed Description	3590
7.539 type_utils.hpp File Reference	3590
7.539.1 Detailed Description	3590
7.540 types_traits.hpp File Reference	3590
7.540.1 Detailed Description	3591
7.541 iterator.hpp File Reference	3591
7.541.1 Detailed Description	3591
7.542 point_iterator.hpp File Reference	3591
7.542.1 Detailed Description	3591
7.543 exception.hpp File Reference	3591
7.543.1 Detailed Description	3592
7.544 hash_policy.hpp File Reference	3592
7.544.1 Detailed Description	3593
7.545 list_update_policy.hpp File Reference	3593
7.545.1 Detailed Description	3593
7.546 priority_queue.hpp File Reference	3593
7.546.1 Detailed Description	3593
7.547 tag_and_trait.hpp File Reference	3593
7.547.1 Detailed Description	3594
7.548 tree_policy.hpp File Reference	3594
7.548.1 Detailed Description	3594
7.549 trie_policy.hpp File Reference	3595
7.549.1 Detailed Description	3595
7.550 pod_char_traits.h File Reference	3595
7.550.1 Detailed Description	3595

7.551 pointer.h File Reference	3596
7.551.1 Detailed Description	3597
7.552 pool_allocator.h File Reference	3597
7.552.1 Detailed Description	3598
7.553 rb_tree File Reference	3598
7.553.1 Detailed Description	3598
7.554 rc_string_base.h File Reference	3598
7.554.1 Detailed Description	3598
7.555 rope File Reference	3599
7.555.1 Detailed Description	3602
7.556 ropeimpl.h File Reference	3602
7.556.1 Detailed Description	3608
7.557 slist File Reference	3608
7.557.1 Detailed Description	3608
7.558 sso_string_base.h File Reference	3609
7.558.1 Detailed Description	3609
7.559 stdio_filebuf.h File Reference	3609
7.559.1 Detailed Description	3609
7.560 stdio_sync_filebuf.h File Reference	3609
7.560.1 Detailed Description	3609
7.561 string_conversions.h File Reference	3609
7.561.1 Detailed Description	3609
7.562 throw_allocator.h File Reference	3610
7.562.1 Detailed Description	3611
7.563 type_traits.h File Reference	3611
7.563.1 Detailed Description	3611
7.564 typelist.h File Reference	3611
7.564.1 Detailed Description	3612
7.565 vstring.h File Reference	3612
7.565.1 Detailed Description	3615
7.566 vstring.tcc File Reference	3615
7.566.1 Detailed Description	3616
7.567 vstring_fwd.h File Reference	3616
7.567.1 Detailed Description	3616
7.568 vstring_util.h File Reference	3616
7.568.1 Detailed Description	3616
7.569 fenv.h File Reference	3617
7.569.1 Detailed Description	3617
7.570 experimental/filesystem File Reference	3617

7.570.1 Detailed Description	3617
7.571 filesystem File Reference	3617
7.571.1 Detailed Description	3617
7.572 flat_map File Reference	3617
7.572.1 Detailed Description	3617
7.573 flat_set File Reference	3617
7.573.1 Detailed Description	3617
7.574 format File Reference	3617
7.574.1 Detailed Description	3617
7.575 debug/forward_list File Reference	3617
7.575.1 Detailed Description	3618
7.576 experimental/forward_list File Reference	3618
7.576.1 Detailed Description	3618
7.577 forward_list File Reference	3619
7.577.1 Detailed Description	3619
7.578 fstream File Reference	3619
7.578.1 Detailed Description	3619
7.579 experimental/functional File Reference	3620
7.579.1 Detailed Description	3620
7.580 ext/functional File Reference	3620
7.580.1 Detailed Description	3621
7.581 functional File Reference	3621
7.581.1 Detailed Description	3623
7.582 future File Reference	3623
7.582.1 Detailed Description	3624
7.583 generator File Reference	3624
7.583.1 Detailed Description	3624
7.584 inplace_vector File Reference	3624
7.584.1 Detailed Description	3624
7.585 iomanip File Reference	3625
7.585.1 Detailed Description	3626
7.586 ios File Reference	3626
7.586.1 Detailed Description	3626
7.587 iosfwd File Reference	3627
7.587.1 Detailed Description	3627
7.588 iostream File Reference	3627
7.588.1 Detailed Description	3627
7.589 istream File Reference	3627
7.589.1 Detailed Description	3628

7.590 experimental/iterator File Reference	3628
7.590.1 Detailed Description	3629
7.591 ext/iterator File Reference	3629
7.591.1 Detailed Description	3629
7.592 iterator File Reference	3629
7.592.1 Detailed Description	3629
7.593 latch File Reference	3629
7.593.1 Detailed Description	3629
7.594 limits File Reference	3629
7.594.1 Detailed Description	3631
7.595 debug/list File Reference	3631
7.595.1 Detailed Description	3631
7.596 experimental/list File Reference	3631
7.596.1 Detailed Description	3632
7.597 list File Reference	3632
7.597.1 Detailed Description	3632
7.598 locale File Reference	3632
7.598.1 Detailed Description	3632
7.599 debug/map File Reference	3632
7.599.1 Detailed Description	3632
7.600 experimental/map File Reference	3632
7.600.1 Detailed Description	3633
7.601 map File Reference	3633
7.601.1 Detailed Description	3633
7.602 math.h File Reference	3633
7.602.1 Detailed Description	3635
7.602.2 Function Documentation	3635
7.603 mdspan File Reference	3637
7.603.1 Detailed Description	3641
7.604 experimental/memory File Reference	3641
7.604.1 Detailed Description	3642
7.605 ext/memory File Reference	3642
7.605.1 Detailed Description	3643
7.606 memory File Reference	3643
7.606.1 Detailed Description	3643
7.607 experimental/memory_resource File Reference	3643
7.607.1 Detailed Description	3643
7.607.2 Function Documentation	3644
7.608 memory_resource File Reference	3644

7.608.1 Detailed Description	3644
7.608.2 Function Documentation	3644
7.609 mutex File Reference	3645
7.609.1 Detailed Description	3645
7.610 numbers File Reference	3645
7.610.1 Detailed Description	3645
7.611 experimental/numeric File Reference	3645
7.611.1 Detailed Description	3645
7.612 ext/numeric File Reference	3646
7.612.1 Detailed Description	3646
7.613 numeric File Reference	3646
7.613.1 Detailed Description	3647
7.614 parallel/numeric File Reference	3647
7.614.1 Detailed Description	3649
7.615 experimental/optional File Reference	3649
7.615.1 Detailed Description	3650
7.616 optional File Reference	3650
7.616.1 Detailed Description	3650
7.617 ostream File Reference	3650
7.617.1 Detailed Description	3650
7.618 algo.h File Reference	3650
7.618.1 Detailed Description	3659
7.619 algobase.h File Reference	3660
7.619.1 Detailed Description	3661
7.620 balanced_quicksort.h File Reference	3662
7.620.1 Detailed Description	3662
7.621 base.h File Reference	3662
7.621.1 Detailed Description	3663
7.622 basic_iterator.h File Reference	3663
7.622.1 Detailed Description	3663
7.623 checkers.h File Reference	3663
7.623.1 Detailed Description	3663
7.624 compiletime_settings.h File Reference	3663
7.624.1 Detailed Description	3664
7.624.2 Macro Definition Documentation	3664
7.625 equally_split.h File Reference	3665
7.625.1 Detailed Description	3665
7.626 features.h File Reference	3665
7.626.1 Detailed Description	3665

7.626.2 Macro Definition Documentation	3665
7.627 find.h File Reference	3667
7.627.1 Detailed Description	3667
7.628 find_selectors.h File Reference	3667
7.628.1 Detailed Description	3668
7.629 for_each.h File Reference	3668
7.629.1 Detailed Description	3668
7.630 for_each_selectors.h File Reference	3668
7.630.1 Detailed Description	3668
7.631 iterator.h File Reference	3669
7.631.1 Detailed Description	3669
7.632 list_partition.h File Reference	3669
7.632.1 Detailed Description	3669
7.633 losertree.h File Reference	3669
7.633.1 Detailed Description	3670
7.634 merge.h File Reference	3670
7.634.1 Detailed Description	3670
7.635 multiseq_selection.h File Reference	3670
7.635.1 Detailed Description	3671
7.636 multiway_merge.h File Reference	3671
7.636.1 Detailed Description	3674
7.636.2 Macro Definition Documentation	3674
7.637 multiway_mergesort.h File Reference	3674
7.637.1 Detailed Description	3674
7.638 numericfwd.h File Reference	3674
7.638.1 Detailed Description	3676
7.639 omp_loop.h File Reference	3676
7.639.1 Detailed Description	3676
7.640 omp_loop_static.h File Reference	3676
7.640.1 Detailed Description	3677
7.641 par_loop.h File Reference	3677
7.641.1 Detailed Description	3677
7.642 parallel.h File Reference	3677
7.642.1 Detailed Description	3677
7.643 partial_sum.h File Reference	3677
7.643.1 Detailed Description	3677
7.644 partition.h File Reference	3678
7.644.1 Detailed Description	3678
7.644.2 Macro Definition Documentation	3678

7.645 queue.h File Reference	3678
7.645.1 Detailed Description	3678
7.645.2 Macro Definition Documentation	3679
7.646 quicksort.h File Reference	3679
7.646.1 Detailed Description	3679
7.647 random_number.h File Reference	3679
7.647.1 Detailed Description	3679
7.648 random_shuffle.h File Reference	3679
7.648.1 Detailed Description	3680
7.649 search.h File Reference	3680
7.649.1 Detailed Description	3680
7.650 set_operations.h File Reference	3680
7.650.1 Detailed Description	3681
7.651 settings.h File Reference	3681
7.651.1 Detailed Description	3681
7.651.2 Deciding whether to run an algorithm in parallel.	3681
7.651.3 Macro Definition Documentation	3682
7.652 sort.h File Reference	3682
7.652.1 Detailed Description	3683
7.653 tags.h File Reference	3683
7.653.1 Detailed Description	3683
7.654 types.h File Reference	3683
7.654.1 Detailed Description	3684
7.655 unique_copy.h File Reference	3684
7.655.1 Detailed Description	3684
7.656 workstealing.h File Reference	3684
7.656.1 Detailed Description	3685
7.657 print File Reference	3685
7.657.1 Detailed Description	3685
7.658 queue File Reference	3685
7.658.1 Detailed Description	3685
7.659 experimental/random File Reference	3685
7.659.1 Detailed Description	3685
7.660 random File Reference	3685
7.660.1 Detailed Description	3685
7.661 ranges File Reference	3686
7.661.1 Detailed Description	3689
7.662 experimental/ratio File Reference	3689
7.662.1 Detailed Description	3689

7.663 ratio File Reference	3689
7.663.1 Detailed Description	3690
7.664 tr2/ratio File Reference	3691
7.664.1 Detailed Description	3691
7.665 experimental/regex File Reference	3691
7.665.1 Detailed Description	3691
7.666 regex File Reference	3691
7.666.1 Detailed Description	3691
7.667 scoped_allocator File Reference	3692
7.667.1 Detailed Description	3692
7.668 semaphore File Reference	3692
7.668.1 Detailed Description	3692
7.669 debug/set File Reference	3692
7.669.1 Detailed Description	3692
7.670 experimental/set File Reference	3692
7.670.1 Detailed Description	3693
7.671 set File Reference	3693
7.671.1 Detailed Description	3693
7.672 shared_mutex File Reference	3693
7.672.1 Detailed Description	3693
7.673 source_location File Reference	3693
7.673.1 Detailed Description	3693
7.674 span File Reference	3693
7.674.1 Detailed Description	3693
7.675 sstream File Reference	3694
7.675.1 Detailed Description	3694
7.676 stack File Reference	3694
7.676.1 Detailed Description	3694
7.677 stdatomic.h File Reference	3694
7.677.1 Detailed Description	3694
7.678 stdbit.h File Reference	3694
7.678.1 Detailed Description	3694
7.679 stdckdint.h File Reference	3695
7.679.1 Detailed Description	3695
7.680 stdexcept File Reference	3695
7.680.1 Detailed Description	3695
7.681 stdlib.h File Reference	3695
7.681.1 Detailed Description	3695
7.681.2 Function Documentation	3695

7.682 stop_token File Reference	3695
7.682.1 Detailed Description	3695
7.683 streambuf File Reference	3695
7.683.1 Detailed Description	3696
7.684 debug/string File Reference	3696
7.684.1 Detailed Description	3698
7.685 experimental/string File Reference	3698
7.685.1 Detailed Description	3699
7.686 string File Reference	3699
7.686.1 Detailed Description	3699
7.687 experimental/string_view File Reference	3699
7.687.1 Detailed Description	3701
7.688 string_view File Reference	3701
7.688.1 Detailed Description	3702
7.689 syncstream File Reference	3702
7.689.1 Detailed Description	3702
7.690 experimental/system_error File Reference	3702
7.690.1 Detailed Description	3703
7.691 system_error File Reference	3703
7.691.1 Detailed Description	3703
7.692 text_encoding File Reference	3703
7.692.1 Detailed Description	3703
7.693 tgmth.h File Reference	3703
7.693.1 Detailed Description	3703
7.694 thread File Reference	3704
7.694.1 Detailed Description	3704
7.695 bool_set File Reference	3704
7.695.1 Detailed Description	3705
7.696 bool_set.tcc File Reference	3705
7.696.1 Detailed Description	3705
7.697 dynamic_bitset File Reference	3705
7.697.1 Detailed Description	3706
7.698 dynamic_bitset.tcc File Reference	3706
7.698.1 Detailed Description	3706
7.699 experimental/tuple File Reference	3706
7.699.1 Detailed Description	3707
7.700 tuple File Reference	3707
7.700.1 Detailed Description	3708
7.701 experimental/type_traits File Reference	3708

7.701.1 Detailed Description	3711
7.702 tr2/type_traits File Reference	3712
7.702.1 Detailed Description	3712
7.703 type_traits File Reference	3712
7.703.1 Detailed Description	3715
7.704 typeindex File Reference	3715
7.704.1 Detailed Description	3715
7.705 debug/unordered_map File Reference	3716
7.705.1 Detailed Description	3718
7.706 experimental/unordered_map File Reference	3718
7.706.1 Detailed Description	3718
7.707 unordered_map File Reference	3718
7.707.1 Detailed Description	3718
7.708 debug/unordered_set File Reference	3719
7.708.1 Detailed Description	3721
7.709 experimental/unordered_set File Reference	3721
7.709.1 Detailed Description	3721
7.710 unordered_set File Reference	3721
7.710.1 Detailed Description	3721
7.711 experimental/utility File Reference	3721
7.711.1 Detailed Description	3722
7.712 utility File Reference	3722
7.712.1 Detailed Description	3722
7.713 valarray File Reference	3722
7.713.1 Detailed Description	3722
7.714 variant File Reference	3722
7.714.1 Detailed Description	3722
7.715 debug/vector File Reference	3722
7.715.1 Detailed Description	3723
7.716 experimental/vector File Reference	3723
7.716.1 Detailed Description	3724
7.717 vector File Reference	3724
7.717.1 Detailed Description	3724
7.718 atomic_word.h File Reference	3724
7.718.1 Detailed Description	3724
7.719 basic_file.h File Reference	3724
7.719.1 Detailed Description	3724
7.720 c++allocator.h File Reference	3724
7.720.1 Detailed Description	3725

7.721 c++config.h File Reference	3725
7.721.1 Detailed Description	3730
7.722 c++io.h File Reference	3731
7.722.1 Detailed Description	3731
7.723 c++locale.h File Reference	3731
7.723.1 Detailed Description	3731
7.724 c++locale_internal.h File Reference	3731
7.724.1 Detailed Description	3731
7.725 parallel/compatibility.h File Reference	3732
7.725.1 Detailed Description	3732
7.726 x86_64-pc-linux-gnu/bits/compatibility.h File Reference	3732
7.726.1 Detailed Description	3732
7.727 cpu_defines.h File Reference	3732
7.727.1 Detailed Description	3732
7.728 ctype_base.h File Reference	3732
7.728.1 Detailed Description	3732
7.729 ctype_inline.h File Reference	3732
7.729.1 Detailed Description	3733
7.730 cxxabi_tweaks.h File Reference	3733
7.730.1 Detailed Description	3733
7.731 error_constants.h File Reference	3733
7.731.1 Detailed Description	3734
7.732 extc++.h File Reference	3734
7.732.1 Detailed Description	3734
7.733 messages_members.h File Reference	3734
7.733.1 Detailed Description	3734
7.734 opt_random.h File Reference	3734
7.734.1 Detailed Description	3734
7.735 os_defines.h File Reference	3734
7.735.1 Detailed Description	3734
7.736 stdc++.h File Reference	3734
7.736.1 Detailed Description	3734
7.737 stdtr1c++.h File Reference	3734
7.737.1 Detailed Description	3734
7.738 time_members.h File Reference	3735
7.738.1 Detailed Description	3735
7.739 compare File Reference	3735
7.739.1 Detailed Description	3736
7.740 cxxabi.h File Reference	3736

7.740.1 Detailed Description	3737
7.740.2 Function Documentation	3737
7.741 exception File Reference	3738
7.741.1 Detailed Description	3738
7.742 initializer_list File Reference	3739
7.742.1 Detailed Description	3739
7.743 new File Reference	3739
7.743.1 Detailed Description	3740
7.743.2 Function Documentation	3740
7.744 typeid File Reference	3740
7.744.1 Detailed Description	3741
Index	3743

1 Deprecated List

Module `negators`

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

Module `pointer_adaptors`

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

Module `ptrmem_adaptors`

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

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

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

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

Deprecated in C++23.

Class `std::auto_ptr<_Tp>`

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

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

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

Struct `std::is_literal_type<_Tp>`

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

Struct `std::is_pod<_Tp>`

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

Struct `std::is_trivial<_Tp>`

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

Struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`

Deprecated since C++17. The recommended alternative is to simply define the typedefs directly in your iterator class.

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

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

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

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

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

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

2 Todo List

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html

Member [__gnu_cxx::power](#) ([_Tp __x](#), [_Integer __n](#), [_MonoidOperation __monoid_op](#))

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::power](#) ([_Tp __x](#), [_Integer __n](#))

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::random_sample](#) ([_InputIterator __first](#), [_InputIterator __last](#), [_RandomAccessIterator __↵out_first](#), [_RandomAccessIterator __out_last](#))

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

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

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Struct [__gnu_cxx::rb_tree](#)<[_Key](#), [_Value](#), [_KeyOfValue](#), [_Compare](#), [_Alloc](#)>

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::rope](#)<[_CharT](#), [_Alloc](#)>

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::slist](#)<[_Tp](#), [_Alloc](#)>

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Module [mathsf](#)

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

Class [std::basic_string](#)<[_CharT](#), [_Traits](#), [_Alloc](#)>

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [std::regex_traits](#)<[_Ch_type](#)>::[transform_primary](#) ([_Fwd_iter __first](#), [_Fwd_iter __last](#)) const

Implement this function correctly.

3 Topic Documentation

3.1 Algorithms

Collaboration diagram for Algorithms:



Topics

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

3.1.1 Detailed Description

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

3.1.2 Generalized Numeric operations

Collaboration diagram for Generalized Numeric operations:



Functions

- `template<typename _InputIterator, typename _Tp>`
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`
`constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`
`constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`
`constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`
`constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`
`constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

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

3.1.2.1 Detailed Description

3.1.2.2 Function Documentation

accumulate() [1/2]

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

Accumulate values in a range.

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

Parameters

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

Returns

The final sum.

accumulate() [2/2]

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

Accumulate values in a range with operation.

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

Parameters

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

Returns

The final sum.

adjacent_difference() [1/2]

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

Return differences between adjacent values.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to result.

adjacent_difference() [2/2]

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

Return differences between adjacent values.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to result.

exclusive_scan() [1/2]

```
template<typename _InputIterator, typename _OutputIterator, typename _Tp>
_OutputIterator std::exclusive_scan (
    _InputIterator __first,
    _InputIterator __last,
```



```

    _OutputIterator __result,
    _Tp __init) [inline], [constexpr]

```

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

Parameters

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

Returns

The end of the output range.

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

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

References [exclusive_scan\(\)](#), and [move\(\)](#).

exclusive_scan() [2/2]

```

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

```

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

Parameters

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

Returns

The end of the output range.

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

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

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

Referenced by [exclusive_scan\(\)](#).

inclusive_scan() [1/3]

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

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

Parameters

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

Returns

The end of the output range.

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

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

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

inclusive_scan() [2/3]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References [inclusive_scan\(\)](#), and [move\(\)](#).

inclusive_scan() [3/3]

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

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

Parameters

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

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

Returns

The end of the output range.

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

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

Referenced by [inclusive_scan\(\)](#), and [inclusive_scan\(\)](#).

`inner_product()` [1/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

inner_product() [2/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

iota()

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

Create a range of sequentially increasing values.

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

Parameters

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

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

partial_sum() [1/2]

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

Return list of partial sums.

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

Parameters

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

Returns

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

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs_pu\(\)](#), and [__gnu_parallel::__sequential_random_shuffle\(\)](#).

partial_sum() [2/2]

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

Return list of partial sums.

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

Parameters

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

Returns

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

reduce() [1/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

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

reduce() [2/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

References [move\(\)](#), and [reduce\(\)](#).

reduce() [3/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

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

Referenced by [reduce\(\)](#), and [reduce\(\)](#).

transform_exclusive_scan()

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

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

Parameters

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

Returns

The end of the output range.

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

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

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

transform_inclusive_scan() [1/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References [move\(\)](#), and [transform_inclusive_scan\(\)](#).

transform_inclusive_scan() [2/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

Referenced by [transform_inclusive_scan\(\)](#).

transform_reduce() [1/3]

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

Transform the elements of a range and reduce.

Parameters

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

Returns

The final sum.

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

transform_reduce() [2/3]

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

Combine elements from two ranges and reduce.

Parameters

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

Returns

The final sum.

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

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

References [move\(\)](#), and [transform_reduce\(\)](#).

transform_reduce() [3/3]

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

Combine elements from two ranges and reduce.

Parameters

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

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

Returns

The final sum.

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

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

Referenced by [transform_reduce\(\)](#).

3.1.3 Mutating

Collaboration diagram for Mutating:



Functions

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

- `template<typename _OutputIterator, typename _Size, typename _Generator>`
`constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI>`
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`
`constexpr _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate>`
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>`
`constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`
`constexpr _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator>`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _ForwardIterator, typename _Tp>`
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`
`constexpr _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`
`constexpr _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`
`constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`
`constexpr _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp>`
`constexpr void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _BidirectionalIterator>`
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator>`
`constexpr _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator>`
`constexpr _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`

- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate>`
`constexpr _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator>`
`constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`
`constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

3.1.3.1 Detailed Description

3.1.3.2 Function Documentation

`copy()`

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

Copies the range `[first,last)` into `result`.

Parameters

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

Returns

result + (last - first)

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

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

copy_backward()

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

Copies the range `[first,last)` into `result`.

Parameters

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

Returns

result - (last - first)

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

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

copy_if()

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

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

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

copy_n()

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

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

Parameters

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

Returns

`result+n`.

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

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

fill()

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

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

Parameters

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

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

fill_n()

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

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

Parameters

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

Returns

The iterator at first+n.

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

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

generate()

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

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

Parameters

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

Returns

`generate()` returns no value.

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

generate_n()

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

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

Parameters

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

Returns

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

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

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

is_partitioned()

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

Checks whether the sequence is partitioned.

Parameters

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

Returns

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

iter_swap()

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

Swaps the contents of two iterators.

Parameters

<code>__a</code>	An iterator.
<code>__b</code>	Another iterator.

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

move()

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

Moves the range [first,last) into result.

Parameters

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

Returns

result + (last - first)

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

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

move_backward()

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

Moves the range [first,last) into result.

Parameters

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

Returns

result - (last - first)

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

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

partition()

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

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

Parameters

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

Returns

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

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

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

partition_copy()

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

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

Parameters

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

Returns

A pair designating the ends of the resulting sequences.

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

partition_point()

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

Find the partition point of a partitioned range.

Parameters

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

Returns

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

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

random_shuffle() [1/2]

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

Randomly shuffle the elements of a sequence.

Parameters

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

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

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

random_shuffle() [2/2]

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

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

Parameters

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

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

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

remove()

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

Remove elements from a sequence.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

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

remove_copy()

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

Copy a sequence, removing elements of a given value.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

remove_copy_if()

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

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

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`. `remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

remove_if()

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

Remove elements from a sequence using a predicate.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

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

replace()

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

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

Parameters

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

Returns

`replace()` returns no value.

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

replace_copy_if()

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

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

Parameters

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

Returns

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

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

replace_if()

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

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

Parameters

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

Returns

`replace_if()` returns no value.

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

reverse()

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

Reverse a sequence.

Parameters

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

Returns

`reverse()` returns no value.

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

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

reverse_copy()

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

Copy a sequence, reversing its elements.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

rotate()

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

Rotate the elements of a sequence.

Parameters

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

Returns

`first + (last - middle).`

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

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

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

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

rotate_copy()

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

Copy a sequence, rotating its elements.

Parameters

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

<code>__result</code>	An output iterator.
-----------------------	---------------------

Returns

An iterator designating the end of the resulting sequence.

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

Returns

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

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

shuffle()

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

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

Parameters

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

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

References [__gen_two_uniform_ints\(\)](#), [std::pair<_T1,_T2>::first](#), and [std::pair<_T1,_T2>::second](#).

stable_partition()

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

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

Parameters

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

Returns

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

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

swap_ranges()

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

Swap the elements of two sequences.

Parameters

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

Returns

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

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

transform() [1/2]

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

Perform an operation on a sequence.

Parameters

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

Returns

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

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

`unary_op` must not alter its argument.

transform() [2/2]

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

Perform an operation on corresponding elements of two sequences.

Parameters

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

<code>__binary_op</code>	A binary operator.
--------------------------	--------------------

Returns

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

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

`binary_op` must not alter either of its arguments.

unique() [1/2]

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

Remove consecutive duplicate values from a sequence.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

unique() [2/2]

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

Remove consecutive values from a sequence using a predicate.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

unique_copy() [1/2]

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

Copy a sequence, removing consecutive duplicate values.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

unique_copy() [2/2]

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

Copy a sequence, removing consecutive values using a predicate.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

3.1.4 Non-Mutating

Collaboration diagram for Non-Mutating:

**Functions**

- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`
`constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`

- `template<typename _InputIterator, typename _Predicate>`
`constexpr bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp>`
`constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _I1, typename _I2>`
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2>`
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`
`constexpr bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp>`
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator>`
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function>`
`constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`
`constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`

- `template<typename _InputIterator1, typename _InputIterator2>`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate>`
`constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp>`
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

3.1.4.1 Detailed Description

3.1.4.2 Function Documentation

`adjacent_find()` [1/2]

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

Find two adjacent values in a sequence that are equal.

Parameters

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

Returns

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

adjacent_find() [2/2]

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

Find two adjacent values in a sequence using a predicate.

Parameters

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

Returns

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

all_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

any_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

count()

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

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

Parameters

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

Returns

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

count_if()

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

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

Parameters

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

Returns

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

equal() [1/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

equal() [2/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

equal() [3/4]

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

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

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

equal() [4/4]

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

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

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

find()

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

Find the first occurrence of a value in a sequence.

Parameters

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

Returns

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

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

find_end() [1/2]

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

Find last matching subsequence in a sequence.

Parameters

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

Returns

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

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

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

find_end() [2/2]

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

Find last matching subsequence in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
-----------------------	---------------------------

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

Returns

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

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

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

`find_first_of()` [1/2]

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

Find element from a set in a sequence.

Parameters

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

Returns

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

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

find_first_of() [2/2]

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

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

Parameters

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

Returns

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

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

find_if()

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

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

Parameters

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

Returns

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

find_if_not()

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

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

Parameters

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

Returns

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

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

for_each()

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

Apply a function to every element of a sequence.

Parameters

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

Returns

`__f`

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

for_each_n()

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

Apply a function to every element of a sequence.

Parameters

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

Returns

`__first+__n`

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

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

is_permutation() [1/4]

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

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

Parameters

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

Returns

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

is_permutation() [2/4]

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

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

Parameters

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

Returns

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

is_permutation() [3/4]

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

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

Parameters

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

Returns

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

is_permutation() [4/4]

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

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

Parameters

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

Returns

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

mismatch() [1/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [2/4]

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

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [3/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [4/4]

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

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

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

none_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

search() [1/2]

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

Search a sequence for a matching sub-sequence.

Parameters

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

Returns

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

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

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

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

search() [2/2]

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

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

Parameters

<code>__first1</code>	A forward iterator.
-----------------------	---------------------

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

Returns

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

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

See also

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

search_n() [1/2]

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

Search a sequence for a number of consecutive values.

Parameters

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

Returns

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

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

search_n() [2/2]

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

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

Parameters

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

Returns

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

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

3.1.5 Sorting

Collaboration diagram for Sorting:



Topics

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

Functions

- `template<typename _Tp>`
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _BidirectionalIterator>`
`constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`
`constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator>`
`constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _II1, typename _II2>`
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare>`
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _InputIter1, typename _InputIter2, typename _Comp>`
`constexpr auto std::lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))`
- `template<typename _Tp>`
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`

- `template<typename _Tp, typename _Compare>`
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator>`
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator>`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator>`
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>`
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator>`
`constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`
`constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

3.1.5.1 Detailed Description

3.1.5.2 Function Documentation

clamp() [1/2]

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

Returns the value clamped between lo and hi.

Parameters

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

Return values

<code>`__lo`</code>	if <code>__val < __lo</code>
<code>`__hi`</code>	if <code>__hi < __val</code>
<code>`__val`</code>	otherwise.

Precondition

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

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

clamp() [2/2]

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

Returns the value clamped between lo and hi.

Parameters

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

Return values

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

Precondition

`__comp(__hi, __lo)` is false.

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

inplace_merge() [1/2]

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

Merges two sorted ranges in place.

Parameters

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

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

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

inplace_merge() [2/2]

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

Merges two sorted ranges in place.

Parameters

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

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

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

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

is_sorted() [1/2]

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

Determines whether the elements of a sequence are sorted.

Parameters

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

Returns

True if the elements are sorted, false otherwise.

is_sorted() [2/2]

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

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

Parameters

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

Returns

True if the elements are sorted, false otherwise.

is_sorted_until() [1/2]

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

Determines the end of a sorted sequence.

Parameters

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

Returns

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

is_sorted_until() [2/2]

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

Determines the end of a sorted sequence using comparison functor.

Parameters

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

Returns

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

lexicographical_compare() [1/2]

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

Performs **dictionary** comparison on ranges.

Parameters

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

Returns

A boolean true or false.

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

lexicographical_compare() [2/2]

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

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A comparison functor .

Returns

A boolean true or false.

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

lexicographical_compare_three_way()

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

Performs dictionary comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A comparison functor .

Returns

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

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

max() [1/2]

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

This does what you think it does.

Parameters

\leftarrow __a	A thing of arbitrary type.
\leftarrow __b	Another thing of arbitrary type.

Returns

The greater of the parameters.

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

Referenced by [__gnu_parallel::__find_template\(\)](#), [__gnu_parallel::__parallel_nth_element\(\)](#), [__gnu_parallel::__parallel_partial_sum_linear\(\)](#), [__gnu_parallel::__parallel_partition\(\)](#), [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#), [__gnu_parallel::__qsb_conquer\(\)](#), [__gnu_parallel::__search_template\(\)](#), [__gnu_parallel::__sequential_random_shuffle\(\)](#), [std::_Deque_base<_Tp, _Alloc>::_M_initialize_buckets\(\)](#), [std::deque<_Tp, _Alloc>::_M_reallocate_map\(\)](#), [clamp\(\)](#), [clamp\(\)](#), [__gnu_parallel::multiseq_partition\(\)](#), [__gnu_parallel::multiseq_select\(\)](#), [std::shuffle_order_engine<_RandomNumberEngine, __k>::operator\(\)](#), [std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow\(\)](#), and [std::philox_engine<_UIntType, __w, __n, __r, __consts>::seed\(\)](#).

max() [2/2]

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

This does what you think it does.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.
__comp	A comparison functor .

Returns

The greater of the parameters.

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

max_element() [1/2]

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

Return the maximum element in a range.

Parameters

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

Returns

Iterator referencing the first instance of the largest value.

max_element() [2/2]

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

Return the maximum element in a range using comparison functor.

Parameters

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

Returns

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

merge() [1/2]

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

Merges two sorted ranges.

Parameters

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

Returns

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

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

merge() [2/2]

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

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
-----------------------	--------------

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

Returns

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

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

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

`min()` [1/2]

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

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The lesser of the parameters.

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

Referenced by [__gnu_parallel::__find_template\(\)](#), [__gnu_parallel::__find_template\(\)](#), [__gnu_parallel::__for_each_template_random_access_iterator_range\(\)](#), [__gnu_parallel::__parallel_partial_sum_linear\(\)](#), [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#), [__gnu_parallel::__parallel_sort_qs_divide_and_conquer\(\)](#), [__gnu_parallel::__qsb_conquer\(\)](#), [__sample\(\)](#), [__gnu_parallel::__search_template\(\)](#), [__gnu_parallel::__sequential_multiway_merge\(\)](#), [__gnu_parallel::__sequential_random_shuffle\(\)](#), [clamp\(\)](#), [clamp\(\)](#), [__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare\(\)](#)

```

__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >
__gnu_cxx::__versa_string<char >::compare(), std::basic_string<_CharT >::compare(), std::basic_string<_CharT >::compare(),
std::basic_string<_CharT >::compare(), std::basic_string<_CharT >::compare(), std::basic_string<_CharT >::compare(),
std::basic_string<_CharT >::compare(), std::basic_string<_CharT >::max_size(), __gnu_parallel::multiseq_partition(),
__gnu_parallel::multiseq_selection(), std::shuffle_order_engine<_RandomNumberEngine, __k >::operator>(),
std::basic_stringbuf<_CharT, _Traits, _Alloc >::overflow(), __gnu_parallel::parallel_multiway_merge(), std::basic_filebuf<_CharT, encod
__gnu_cxx::random_sample_n(), __gnu_cxx::random_sample_n(), std::basic_istream<_CharT, _Traits >::readsome(),
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind(), std::basic_string<_CharT, _Traits, _Alloc >::rfind(),
std::basic_streambuf<_CharT, _Traits >::xsgetn(), and std::basic_streambuf<_CharT, _Traits >::xsputn().

```

min() [2/2]

```

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

```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The lesser of the parameters.

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

min_element() [1/2]

```

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

```

Return the minimum element in a range.

Parameters

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

Returns

Iterator referencing the first instance of the smallest value.

min_element() [2/2]

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

Return the minimum element in a range using comparison functor.

Parameters

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

Returns

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

minmax() [1/2]

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

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
------------------	----------------------------

<code>__b</code>	Another thing of arbitrary type.
------------------	----------------------------------

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

minmax() [2/2]

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

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

minmax_element() [1/2]

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

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

Parameters

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

Returns

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

minmax_element() [2/2]

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

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

Parameters

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

Returns

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

next_permutation() [1/2]

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

Permute range into the next *dictionary* ordering.

Parameters

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

Returns

False if wrapped to first permutation, true otherwise.

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

next_permutation() [2/2]

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

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

Parameters

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

Returns

False if wrapped to first permutation, true otherwise.

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

nth_element() [1/2]

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

Sort a sequence just enough to find a particular position.

Parameters

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

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

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

nth_element() [2/2]

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

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

Parameters

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

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

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

partial_sort() [1/2]

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

Sort the smallest elements of a sequence.

Parameters

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

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

partial_sort() [2/2]

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

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

Parameters

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

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

partial_sort_copy() [1/2]

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

Copy the smallest elements of a sequence.

Parameters

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

Returns

An iterator indicating the end of the resulting sequence.

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

partial_sort_copy() [2/2]

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

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

Parameters

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

Returns

An iterator indicating the end of the resulting sequence.

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

prev_permutation() [1/2]

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

Permute range into the previous *dictionary* ordering.

Parameters

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

Returns

False if wrapped to last permutation, true otherwise.

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

prev_permutation() [2/2]

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

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

Parameters

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

Returns

False if wrapped to last permutation, true otherwise.

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

sort() [1/2]

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

Sort the elements of a sequence.

Parameters

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

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

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

sort() [2/2]

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

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

Parameters

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

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

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

stable_sort() [1/2]

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

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

Parameters

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

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

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

stable_sort() [2/2]

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

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

Parameters

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

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

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

3.1.5.3 Binary Search

Collaboration diagram for Binary Search:

**Functions**

- `template<typename _ForwardIterator, typename _Tp>`
`constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`
`constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp>`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp>`
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp>`
`constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`
`constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

3.1.5.3.1 Detailed Description

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

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

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

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

3.1.5.3.2 Function Documentation

binary_search() [1/2]

```
template<typename _ForwardIterator, typename _Tp>
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp &__val) [nodiscard], [constexpr]
```

Determines whether an element exists in a range.

Parameters

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

<code>__val</code>	The search term.
--------------------	------------------

Returns

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

binary_search() [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [nodiscard], [constexpr]
```

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

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

equal_range() [1/2]

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

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

equal_range() [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
pair< _ForwardIterator, _ForwardIterator > std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

lower_bound() [1/2]

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

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

lower_bound() [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
_FForwardIterator std::lower_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the first position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator to the start of a sorted range.
<code>__last</code>	A past-the-end iterator for the sorted range.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

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

upper_bound() [1/2]

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

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

upper_bound() [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
_FForwardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

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

3.1.5.4 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator>`
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

3.1.5.4.1 Detailed Description

3.1.5.4.2 Function Documentation

is_heap() [1/2]

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

Determines whether a range is a heap.

Parameters

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

Returns

True if range is a heap, false otherwise.

is_heap() [2/2]

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

Determines whether a range is a heap using comparison functor.

Parameters

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

Returns

True if range is a heap, false otherwise.

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

is_heap_until() [1/2]

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

Search the end of a heap.

Parameters

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

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [`__first`, `__last`) for which the range [`__first`, *i*) is a heap.

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

is_heap_until() [2/2]

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

Search the end of a heap using comparison functor.

Parameters

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

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [`__first`, `__last`) for which the range [`__first`, *i*) is a heap. Comparisons are made using `__comp`.

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

make_heap() [1/2]

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

Construct a heap over a range.

Parameters

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

This operation makes the elements in [`__first`,`__last`) into a heap.

make_heap() [2/2]

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

Construct a heap over a range using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in [`__first`,`__last`) into a heap. Comparisons are made using `__comp`.

pop_heap() [1/2]

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

Pop an element off a heap.

Parameters

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

Precondition

`[__first, __last)` is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap.

pop_heap() [2/2]

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

Pop an element off a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap. Comparisons are made using `comp`.

push_heap() [1/2]

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

Push an element onto a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first, __last-1)`. After completion, `[__first, __last)` is a valid heap.

push_heap() [2/2]

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

Push an element onto a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap. Compare operations are performed using `comp`.

sort_heap() [1/2]

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

Sort a heap.

Parameters

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

This operation sorts the valid heap in the range `[__first,__last)`.

sort_heap() [2/2]

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

Sort a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation sorts the valid heap in the range `[__first,__last)`. Comparisons are made using `__comp`.

3.1.5.5 Set Operations

Collaboration diagram for Set Operations:

**Functions**

- `template<typename _InputIterator1, typename _InputIterator2>`
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`
`constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`
`constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`
`constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`
`constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`
`constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`
`constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`
`constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`
`constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

3.1.5.5.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

3.1.5.5.2 Function Documentation

includes() [1/2]

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

Determines whether all elements of a sequence exists in a range.

Parameters

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

Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

includes() [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _Compare __comp) [inline], [nodiscard], [constexpr]
```

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)` according to `comp`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`, using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

set_difference() [1/2]

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

Return the difference of two sorted ranges.

Parameters

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

<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_difference() [2/2]

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

Return the difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_intersection() [1/2]

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

Return the intersection of two sorted ranges.

Parameters

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

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_intersection() [2/2]

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

Return the intersection of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
-----------------------	-----------------------

<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_symmetric_difference() [1/2]

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

Return the symmetric difference of two sorted ranges.

Parameters

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

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_symmetric_difference() [2/2]

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

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_union() [1/2]

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

Return the union of two sorted ranges.

Parameters

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

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_union() [2/2]

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

Return the union of two sorted ranges using a comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

3.2 Atomics

Classes

- struct `std::atomic< _Tp >`
- struct `std::atomic< _Tp * >`
- struct `std::atomic< bool >`
- struct `std::atomic< char >`
- struct `std::atomic< char16_t >`
- struct `std::atomic< char32_t >`
- struct `std::atomic< int >`
- struct `std::atomic< long >`
- struct `std::atomic< long long >`
- struct `std::atomic< short >`
- struct `std::atomic< signed char >`
- struct `std::atomic< unsigned char >`
- struct `std::atomic< unsigned int >`
- struct `std::atomic< unsigned long >`
- struct `std::atomic< unsigned long long >`
- struct `std::atomic< unsigned short >`
- struct `std::atomic< wchar_t >`
- struct `std::atomic_flag`

Macros

- `#define ATOMIC_BOOL_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_VAR_INIT(_VI)`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

Typedefs

- typedef unsigned char `std::__atomic_flag_data_type`
- typedef `atomic< bool >` `std::atomic_bool`
- typedef `atomic< char >` `std::atomic_char`
- typedef `atomic< char16_t >` `std::atomic_char16_t`
- typedef `atomic< char32_t >` `std::atomic_char32_t`
- typedef `atomic< int >` `std::atomic_int`
- typedef `atomic< int16_t >` `std::atomic_int16_t`
- typedef `atomic< int32_t >` `std::atomic_int32_t`
- typedef `atomic< int64_t >` `std::atomic_int64_t`

- typedef `atomic< int8_t > std::atomic_int8_t`
- typedef `atomic< int_fast16_t > std::atomic_int_fast16_t`
- typedef `atomic< int_fast32_t > std::atomic_int_fast32_t`
- typedef `atomic< int_fast64_t > std::atomic_int_fast64_t`
- typedef `atomic< int_fast8_t > std::atomic_int_fast8_t`
- typedef `atomic< int_least16_t > std::atomic_int_least16_t`
- typedef `atomic< int_least32_t > std::atomic_int_least32_t`
- typedef `atomic< int_least64_t > std::atomic_int_least64_t`
- typedef `atomic< int_least8_t > std::atomic_int_least8_t`
- typedef `atomic< intmax_t > std::atomic_intmax_t`
- typedef `atomic< intptr_t > std::atomic_intptr_t`
- typedef `atomic< long long > std::atomic_llong`
- typedef `atomic< long > std::atomic_long`
- typedef `atomic< ptrdiff_t > std::atomic_ptrdiff_t`
- typedef `atomic< signed char > std::atomic_schar`
- typedef `atomic< short > std::atomic_short`
- typedef `atomic< size_t > std::atomic_size_t`
- typedef `atomic< unsigned char > std::atomic_uchar`
- typedef `atomic< unsigned int > std::atomic_uint`
- typedef `atomic< uint16_t > std::atomic_uint16_t`
- typedef `atomic< uint32_t > std::atomic_uint32_t`
- typedef `atomic< uint64_t > std::atomic_uint64_t`
- typedef `atomic< uint8_t > std::atomic_uint8_t`
- typedef `atomic< uint_fast16_t > std::atomic_uint_fast16_t`
- typedef `atomic< uint_fast32_t > std::atomic_uint_fast32_t`
- typedef `atomic< uint_fast64_t > std::atomic_uint_fast64_t`
- typedef `atomic< uint_fast8_t > std::atomic_uint_fast8_t`
- typedef `atomic< uint_least16_t > std::atomic_uint_least16_t`
- typedef `atomic< uint_least32_t > std::atomic_uint_least32_t`
- typedef `atomic< uint_least64_t > std::atomic_uint_least64_t`
- typedef `atomic< uint_least8_t > std::atomic_uint_least8_t`
- typedef `atomic< uintmax_t > std::atomic_uintmax_t`
- typedef `atomic< uintptr_t > std::atomic_uintptr_t`
- typedef `atomic< unsigned long long > std::atomic_ullong`
- typedef `atomic< unsigned long > std::atomic_ulong`
- typedef `atomic< unsigned short > std::atomic_ushort`
- typedef `atomic< wchar_t > std::atomic_wchar_t`

Enumerations

- enum class `std::memory_order` : int {
 relaxed , **consume** , **acquire** , **release** ,
 acq_rel , **seq_cst** }

Functions

- `template<typename _ITp>`
`bool std::atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_weak (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_exchange (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_add (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_add (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_add_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_add_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`

- `template<typename _ITp>`
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`
`void std::atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`void std::atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`

- `template<typename _ITp>`
`_ITp std::atomic_load_explicit (const atomic<_ITp> *__a, memory_order __m) noexcept`
- `template<typename _ITp>`
`_ITp std::atomic_load_explicit (const volatile atomic<_ITp> *__a, memory_order __m) noexcept`
- `void std::atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp>`
`void std::atomic_store (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`
`void std::atomic_store (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`
`void std::atomic_store_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`
`void std::atomic_store_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `void std::atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Tp>`
`_Tp std::kill_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod) noexcept`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod) noexcept`

Variables

- `constexpr memory_order std::memory_order_acq_rel`
- `constexpr memory_order std::memory_order_acquire`
- `constexpr memory_order std::memory_order_consume`
- `constexpr memory_order std::memory_order_relaxed`
- `constexpr memory_order std::memory_order_release`
- `constexpr memory_order std::memory_order_seq_cst`

3.2.1 Detailed Description

Components for performing atomic operations.

3.2.2 Macro Definition Documentation

ATOMIC_BOOL_LOCK_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

3.2.3 Typedef Documentation

atomic_bool

```
typedef atomic<bool> std::atomic_bool
```

`atomic_bool`

atomic_char

```
typedef atomic<char> std::atomic_char
```

`atomic_char`

atomic_char16_t

```
typedef atomic<char16_t> std::atomic_char16_t
```

`atomic_char16_t`

atomic_char32_t

```
typedef atomic<char32_t> std::atomic_char32_t
```

`atomic_char32_t`

atomic_int

```
typedef atomic<int> std::atomic_int
```

`atomic_int`

atomic_int16_t

```
typedef atomic<int16_t> std::atomic_int16_t
```

`atomic_int16_t`

atomic_int32_t

```
typedef atomic<int32_t> std::atomic_int32_t
```

`atomic_int32_t`

atomic_int64_t

```
typedef atomic<int64_t> std::atomic_int64_t
```

atomic_int64_t

atomic_int8_t

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic_int8_t

atomic_int_fast16_t

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic_int_fast16_t

atomic_int_fast32_t

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

atomic_int_fast32_t

atomic_int_fast64_t

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

atomic_int_fast64_t

atomic_int_fast8_t

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

atomic_int_fast8_t

atomic_int_least16_t

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

atomic_int_least16_t

atomic_int_least32_t

```
typedef atomic<int_least32_t> std::atomic\_int\_least32\_t
```

[atomic_int_least32_t](#)

atomic_int_least64_t

```
typedef atomic<int_least64_t> std::atomic\_int\_least64\_t
```

[atomic_int_least64_t](#)

atomic_int_least8_t

```
typedef atomic<int_least8_t> std::atomic\_int\_least8\_t
```

[atomic_int_least8_t](#)

atomic_intmax_t

```
typedef atomic<intmax_t> std::atomic\_intmax\_t
```

[atomic_intmax_t](#)

atomic_intptr_t

```
typedef atomic<intptr_t> std::atomic\_intptr\_t
```

[atomic_intptr_t](#)

atomic_llong

```
typedef atomic<long long> std::atomic\_llong
```

[atomic_llong](#)

atomic_long

```
typedef atomic<long> std::atomic\_long
```

[atomic_long](#)

atomic_ptrdiff_t

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic_ptrdiff_t

atomic_schar

```
typedef atomic<signed char> std::atomic_schar
```

atomic_schar

atomic_short

```
typedef atomic<short> std::atomic_short
```

atomic_short

atomic_size_t

```
typedef atomic<size_t> std::atomic_size_t
```

atomic_size_t

atomic_uchar

```
typedef atomic<unsigned char> std::atomic_uchar
```

atomic_uchar

atomic_uint

```
typedef atomic<unsigned int> std::atomic_uint
```

atomic_uint

atomic_uint16_t

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

atomic_uint16_t

atomic_uint32_t

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

atomic_uint32_t

atomic_uint64_t

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

atomic_uint64_t

atomic_uint8_t

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

atomic_uint8_t

atomic_uint_fast16_t

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

atomic_uint_fast16_t

atomic_uint_fast32_t

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

atomic_uint_fast32_t

atomic_uint_fast64_t

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

atomic_uint_fast64_t

atomic_uint_fast8_t

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

atomic_uint_fast8_t

atomic_uint_least16_t

```
typedef atomic<uint_least16_t> std::atomic_uint_least16_t
```

`atomic_uint_least16_t`

atomic_uint_least32_t

```
typedef atomic<uint_least32_t> std::atomic_uint_least32_t
```

`atomic_uint_least32_t`

atomic_uint_least64_t

```
typedef atomic<uint_least64_t> std::atomic_uint_least64_t
```

`atomic_uint_least64_t`

atomic_uint_least8_t

```
typedef atomic<uint_least8_t> std::atomic_uint_least8_t
```

`atomic_uint_least8_t`

atomic_uintmax_t

```
typedef atomic<uintmax_t> std::atomic_uintmax_t
```

`atomic_uintmax_t`

atomic_uintptr_t

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

`atomic_uintptr_t`

atomic_ullong

```
typedef atomic<unsigned long long> std::atomic_ullong
```

`atomic_ullong`

atomic_ulong

```
typedef atomic<unsigned long> std::atomic_ulong
```

`atomic_ulong`

atomic_ushort

```
typedef atomic<unsigned short> std::atomic_ushort
```

`atomic_ushort`

atomic_wchar_t

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

`atomic_wchar_t`

3.2.4 Enumeration Type Documentation**memory_order**

```
enum class std::memory_order : int [strong]
```

Enumeration for `memory_order`.

3.2.5 Function Documentation**kill_dependency()**

```
template<typename _Tp>  
_Tp std::kill_dependency (  
    _Tp __y) [inline], [noexcept]
```

`kill_dependency`

3.3 Concurrency

Collaboration diagram for Concurrency:



Topics

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

3.3.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

3.3.2 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class `std::condition_variable`
- class `std::condition_variable_any`

Enumerations

- enum class `std::cv_status` { `no_timeout` , `timeout` }

Functions

- void `std::notify_all_at_thread_exit` (`condition_variable` &, `unique_lock`< `mutex` >)

3.3.2.1 Detailed Description

Classes for `condition_variable` support.

3.3.2.2 Enumeration Type Documentation

`cv_status`

```
enum class std::cv_status [strong]
```

`cv_status`

3.3.3 Futures

Collaboration diagram for Futures:



Classes

- class `std::__basic_future<_Res>`
- class `std::future<_Res>`
- class `std::future<_Res &>`
- class `std::future<void>`
- class `std::future_error`
- struct `std::is_error_code_enum<future_errc>`
- class `std::packaged_task<_Res(_ArgTypes...)>`
- class `std::promise<_Res>`
- class `std::promise<_Res &>`
- class `std::promise<void>`
- class `std::shared_future<_Res>`
- class `std::shared_future<_Res &>`
- class `std::shared_future<void>`

Enumerations

- enum class `std::future_errc` { `future_already_retrieved` , `promise_already_satisfied` , `no_state` , `broken_promise` }
- enum class `std::future_status` { `ready` , `timeout` , `deferred` }
- enum class `std::launch` { `async` , `deferred` }

Functions

- `std::__basic_future<_Res>::__basic_future` (const `shared_future<_Res>` &) noexcept
- `std::__basic_future<_Res>::__basic_future` (`future<_Res>` &&) noexcept
- `std::__basic_future<_Res>::__basic_future` (`shared_future<_Res>` &&) noexcept
- template<typename _Fn, typename... _Args>
`future<__async_result_of<_Fn, _Args...>> std::async` (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
`future<__async_result_of<_Fn, _Args...>> std::async` (`launch` __policy, _Fn &&__fn, _Args &&... __args)
- const `error_category` & `std::future_category` () noexcept
- `error_code` `std::make_error_code` (`future_errc` __errc) noexcept
- `error_condition` `std::make_error_condition` (`future_errc` __errc) noexcept
- constexpr `launch` `std::operator&` (`launch` __x, `launch` __y) noexcept
- constexpr `launch` & `std::operator&=` (`launch` &__x, `launch` __y) noexcept
- constexpr `launch` `std::operator^` (`launch` __x, `launch` __y) noexcept
- constexpr `launch` & `std::operator^=` (`launch` &__x, `launch` __y) noexcept
- constexpr `launch` `std::operator|` (`launch` __x, `launch` __y) noexcept
- constexpr `launch` & `std::operator|=` (`launch` &__x, `launch` __y) noexcept
- constexpr `launch` `std::operator~` (`launch` __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
`std::packaged_task` (_Fun) -> `packaged_task<_Signature>`
- template<typename _Res, typename... _ArgTypes>
`std::packaged_task` (_Res*)(_ArgTypes...) -> `packaged_task<_Res(_ArgTypes...)>`
- `shared_future<_Res>` > `std::future<_Res>::share` () noexcept
- `shared_future<_Res &>` > `std::future<_Res &>::share` () noexcept
- `shared_future<void>` > `std::future<void>::share` () noexcept
- template<typename _Res, typename... _ArgTypes>
void `std::swap` (`packaged_task<_Res(_ArgTypes...)>` &__x, `packaged_task<_Res(_ArgTypes...)>` &__y) noexcept
- template<typename _Res>
void `std::swap` (`promise<_Res>` &__x, `promise<_Res>` &__y) noexcept

3.3.3.1 Detailed Description

Futures and promises provide support for retrieving the result from an asynchronous function, e.g. one that is running in another thread. A `std::future` represents an asynchronous result that will become ready at some later time. A consumer can wait on a future until the result is ready to be accessed.

Since

C++11

3.3.3.2 Enumeration Type Documentation

future_errc

```
enum class std::future_errc [strong]
```

Error code for futures.

future_status

```
enum class std::future_status [strong]
```

Status code for futures.

launch

```
enum class std::launch [strong]
```

Launch code for futures.

3.3.3.3 Function Documentation

async() [1/2]

```
template<typename _Fn, typename... _Args>
future< __async_result_of< _Fn, _Args... > > std::async (
    _Fn && __fn,
    _Args &&... __args) [inline], [nodiscard]
```

async, potential overload

References [async\(\)](#), and [forward\(\)](#).

async() [2/2]

```
template<typename _Fn, typename... _Args>
future< __async_result_of< _Fn, _Args... > > std::async (
    launch __policy,
    _Fn && __fn,
    _Args &&... __args) [nodiscard]
```

async

References [async\(\)](#), [forward\(\)](#), and [move\(\)](#).

Referenced by [async\(\)](#), and [async\(\)](#).

future_category()

```
const error_category & std::future_category () [noexcept]
```

Points to a statically-allocated object derived from `error_category`.

Referenced by [make_error_code\(\)](#), and [make_error_condition\(\)](#).

make_error_code()

```
error_code std::make_error_code (
    future_errc __errc) [inline], [noexcept]
```

Overload of `make_error_code` for `future_errc`.

References [future_category\(\)](#), and [make_error_code\(\)](#).

Referenced by [make_error_code\(\)](#), [system_category\(\)](#), and [std::filesystem::filesystem_error::what\(\)](#).

make_error_condition()

```
error_condition std::make_error_condition (
    future_errc __errc) [inline], [noexcept]
```

Overload of `make_error_condition` for `future_errc`.

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

Referenced by [system_category\(\)](#).

swap()

```
template<typename _Res, typename... _ArgTypes>
void std::swap (
    packaged_task< _Res(_ArgTypes...)> & __x,
    packaged_task< _Res(_ArgTypes...)> & __y) [inline], [noexcept]
```

swap

3.3.4 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct [std::adopt_lock_t](#)
- struct [std::defer_lock_t](#)
- class [std::lock_guard< _Mutex >](#)
- class [std::mutex](#)
- struct [std::once_flag](#)
- class [std::recursive_mutex](#)
- class [std::recursive_timed_mutex](#)
- class [std::shared_lock< _Mutex >](#)
- class [std::shared_timed_mutex](#)
- class [std::timed_mutex](#)
- struct [std::try_to_lock_t](#)
- class [std::unique_lock< _Mutex >](#)

Functions

- void [std::__once_proxy](#) (void)
- template<typename _Callable, typename... _Args>
void [std::call_once](#) ([once_flag](#) &__once, _Callable &&__f, _Args &&... __args)
- template<typename _L1, typename _L2, typename... _L3>
void [std::lock](#) (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _L1, typename _L2, typename... _L3>
int [std::try_lock](#) (_L1 &__l1, _L2 &__l2, _L3 &... __l3)

Variables

- constexpr [adopt_lock_t](#) std::adopt_lock
- constexpr [defer_lock_t](#) std::defer_lock
- constexpr [try_to_lock_t](#) std::try_to_lock

3.3.4.1 Detailed Description

Classes for mutex support.

3.3.4.2 Function Documentation

call_once()

```
template<typename _Callable, typename... _Args>
void std::call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args)
```

Invoke a callable and synchronize with other calls using the same flag.

lock()

```
template<typename _L1, typename _L2, typename... _L3>
void std::lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3)
```

Generic lock.

Parameters

 _l1	Meets Lockable requirements (try_lock() may throw).
 _l2	Meets Lockable requirements (try_lock() may throw).
 _l3	Meets Lockable requirements (try_lock() may throw).

Exceptions

<i>An</i>	exception thrown by an argument's lock() or try_lock() member.
-----------	--

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to `lock()`, `try_lock()` and `unlock()`. If this function exits via an exception any locks that were obtained will be released.

References [defer_lock](#), and [try_lock\(\)](#).

try_lock()

```
template<typename _L1, typename _L2, typename... _L3>
int std::try_lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3) [inline], [nodiscard]
```

Generic `try_lock`.

Parameters

<code>__l1</code>	Meets Lockable requirements (<code>try_lock()</code> may throw).
<code>__l2</code>	Meets Lockable requirements (<code>try_lock()</code> may throw).
<code>__l3</code>	Meets Lockable requirements (<code>try_lock()</code> may throw).

Returns

Returns -1 if all `try_lock()` calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls `try_lock()` on each argument.

Referenced by [lock\(\)](#).

3.3.4.3 Variable Documentation**adopt_lock**

```
adopt_lock_t std::adopt_lock [inline], [constexpr]
```

Tag used to make a scoped lock take ownership of a locked mutex.

defer_lock

```
defer_lock_t std::defer_lock [inline], [constexpr]
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

Referenced by [lock\(\)](#).

try_to_lock

```
try_to_lock_t std::try_to_lock [inline], [constexpr]
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

3.3.5 Threads

Collaboration diagram for Threads:

**Namespaces**

- namespace [std::this_thread](#)

Classes

- struct [std::hash< thread::id >](#)
- class [std::thread](#)

Functions

- bool **operator==** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept
- strong_ordering **operator<=>** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- template<class _CharT, class _Traits>
[basic_ostream](#)< _CharT, _Traits > & **operator<<** ([basic_ostream](#)< _CharT, _Traits > &__out, [thread::id](#) __id)

3.3.5.1 Detailed Description

Since

C++11

Classes for thread support.

3.4 Containers

Collaboration diagram for Containers:



Topics

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

3.4.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

3.4.2 Associative

Collaboration diagram for Associative:



Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc>`
- class `std::multimap<_Key, _Tp, _Compare, _Alloc>`
- class `std::multiset<_Key, _Compare, _Alloc>`
- class `std::set<_Key, _Compare, _Alloc>`

3.4.2.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

3.4.3 Sequences

Collaboration diagram for Sequences:



Files

- file [inplace_vector](#)

Classes

- struct [std::array<_Tp, _Nm>](#)
- class [std::basic_string<_CharT, _Traits, _Alloc>](#)
- class [std::basic_string_view<_CharT, _Traits>](#)
- class [std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>](#)
- class [std::deque<_Tp, _Alloc>](#)
- class [std::forward_list<_Tp, _Alloc>](#)
- class [std::list<_Tp, _Alloc>](#)
- class [std::priority_queue<_Tp, _Sequence, _Compare>](#)
- class [std::queue<_Tp, _Sequence>](#)
- class [std::stack<_Tp, _Sequence>](#)
- class [std::vector<_Tp, _Alloc>](#)
- class [std::vector<bool, _Alloc>](#)

3.4.3.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

3.4.4 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

3.4.4.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

3.5 Diagnostics

Collaboration diagram for Diagnostics:



Topics

- [Exceptions](#)

Classes

- class `std::error_category`
- class `std::error_code`
- class `std::error_condition`
- struct `std::is_error_code_enum<_Tp>`
- struct `std::is_error_condition_enum<_Tp>`

Functions

- `const error_category & std::generic_category ()` noexcept
- `error_code make_error_code (errc __e)` noexcept
- `error_condition make_error_condition (errc __e)` noexcept
- `template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > & __os, const error_code & __e)`
- `strong_ordering operator<=> (const error_code & __lhs, const error_code & __rhs)` noexcept
- `strong_ordering operator<=> (const error_condition & __lhs, const error_condition & __rhs)` noexcept
- `bool operator== (const error_code & __lhs, const error_code & __rhs)` noexcept
- `bool operator== (const error_code & __lhs, const error_condition & __rhs)` noexcept
- `bool operator== (const error_condition & __lhs, const error_condition & __rhs)` noexcept
- `const error_category & std::system_category ()` noexcept

Variables

- `template<typename _Tp>
constexpr bool std::is_error_code_enum_v`
- `template<typename _Tp>
constexpr bool std::is_error_condition_enum_v`

3.5.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

3.5.2 Function Documentation

`generic_category()`

```
const error\_category & std::generic_category () [noexcept]
```

Error category for `errno` error codes.

Referenced by `std::error_condition::error_condition()`, `std::error_condition::clear()`, `std::error_code::make_error_code()`, and `std::error_condition::make_error_condition()`.

`make_error_code()`

```
error_code make_error_code (  
    errc __e) [related]
```

Create an `error_code` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

References `std::generic_category()`.

Referenced by `error_code()`.

make_error_condition()

```
error_condition make_error_condition (
    errc __e) [related]
```

Create an `error_condition` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

References [error_condition\(\)](#), and [std::generic_category\(\)](#).

Referenced by [error_condition\(\)](#).

operator<<()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const error_code & __e) [related]
```

Write a `std::error_code` to an ostream.

Since

C++11

References [operator<=>\(\)](#).

operator<=>() [1/2]

```
strong_ordering operator<=> (
    const error_code & __lhs,
    const error_code & __rhs) [related]
```

Ordered comparison for `std::error_code`.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

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

operator<=>() [2/2]

```
strong_ordering operator<=> (
    const error_condition & __lhs,
    const error_condition & __rhs) [related]
```

Ordered comparison for `std::error_condition`.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

References [error_condition\(\)](#), and [operator<=>\(\)](#).

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

operator==() [1/3]

```
bool operator== (
    const error_code & __lhs,
    const error_code & __rhs) [related]
```

Equality comparison for `std::error_code`.

Returns true only if they have the same category and the same value.

Since

C++11

operator==() [2/3]

```
bool operator== (
    const error_code & __lhs,
    const error_condition & __rhs) [related]
```

Equality comparison for `std::error_code` and `std::error_condition`.

Uses each category's `equivalent` member function to check whether the values correspond to an equivalent error in that category.

Since

C++11

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

operator==() [3/3]

```
bool operator== (
    const error_condition & __lhs,
    const error_condition & __rhs) [related]
```

Equality comparison for `std::error_condition`.

Returns true only if they have the same category and the same value.

Since

C++11

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

system_category()

```
const error_category & std::system_category () [noexcept]
```

Error category for other error codes defined by the OS.

References [make_error_code\(\)](#), [make_error_condition\(\)](#), and [system_category\(\)](#).

Referenced by [system_category\(\)](#).

3.5.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- class [__cxxabiv1::__forced_unwind](#)
- class [std::bad_alloc](#)
- class [std::experimental::fundamentals_v1::bad_any_cast](#)
- class [std::bad_cast](#)
- class [std::bad_exception](#)
- class [std::bad_function_call](#)
- class [std::experimental::fundamentals_v1::bad_optional_access](#)
- class [std::bad_typeid](#)
- class [std::bad_weak_ptr](#)
- class [std::domain_error](#)
- class [std::exception](#)
- class [std::__unspecified__::exception_ptr](#)
- class [std::ios_base::failure](#)
- struct [__gnu_cxx::forced_error](#)
- class [std::future_error](#)
- class [std::invalid_argument](#)
- class [std::length_error](#)
- class [std::logic_error](#)
- class [std::nested_exception](#)
- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [__gnu_cxx::recursive_init_error](#)
- class [std::regex_error](#)
- class [std::runtime_error](#)
- class [std::system_error](#)
- class [std::underflow_error](#)

Typedefs

- typedef void(* [std::terminate_handler](#)) ()

Functions

- void [__gnu_cxx::__verbose_terminate_handler](#) ()
- [exception_ptr](#) [std::current_exception](#) () noexcept
- [terminate_handler](#) [std::get_terminate](#) () noexcept
- template<typename [_Ex](#)>
constexpr [exception_ptr](#) [std::make_exception_ptr](#) ([_Ex](#) __ex) noexcept
- void [std::rethrow_exception](#) ([exception_ptr](#))
- template<typename [_Ex](#)>
void [std::rethrow_if_nested](#) (const [_Ex](#) &__ex)
- [terminate_handler](#) [std::set_terminate](#) ([terminate_handler](#)) noexcept
- void [std::terminate](#) () noexcept
- template<typename [_Tp](#)>
void [std::throw_with_nested](#) ([_Tp](#) &&__t)
- bool [std::uncaught_exception](#) () noexcept

3.5.3.1 Detailed Description

Since

C++98

Classes and functions for reporting errors via exceptions.

3.5.3.2 Typedef Documentation

terminate_handler

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

3.5.3.3 Function Documentation

__verbose_terminate_handler()

```
void __gnu_cxx::__verbose_terminate_handler ()
```

A replacement for the standard `terminate_handler` which prints more information about the terminating exception (if any) on `stderr`.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.html>.

In 3.4 and later, this is on by default.

current_exception()

```
exception_ptr std::current_exception () [noexcept]
```

Obtain an `exception_ptr` to the currently handled exception.

If there is none, or the currently handled exception is foreign, return the null value.

Since

C++11

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

Referenced by [std::nested_exception::nested_exception\(\)](#), and [make_exception_ptr\(\)](#).

get_terminate()

```
terminate_handler std::get_terminate () [noexcept]
```

Return the current terminate handler.

make_exception_ptr()

```
template<typename _Ex>
exception_ptr std::make_exception_ptr (
    _Ex __ex) [constexpr], [noexcept]
```

Obtain an exception_ptr pointing to a copy of the supplied object.

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

Referenced by [current_exception\(\)](#).

rethrow_exception()

```
void std::rethrow_exception (
    exception_ptr )
```

Throw the object pointed to by the exception_ptr.

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

Referenced by [std::__basic_future<_Res & >::_M_get_result\(\)](#), [rethrow_exception\(\)](#), and [std::nested_exception::rethrow_nested\(\)](#).

rethrow_if_nested()

```
template<typename _Ex>
void std::rethrow_if_nested (
    const _Ex & __ex) [inline]
```

Rethrow a nested exception

If `__ex` contains a `std::nested_exception` object, call its `rethrow_nested()` member to rethrow the stored exception.

After catching an exception thrown by a call to `std::throw_with_nested` this function can be used to rethrow the exception that was active when `std::throw_with_nested` was called.

Since

C++11

set_terminate()

```
terminate_handler std::set_terminate (
    terminate_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

terminate()

```
void std::terminate () [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

Referenced by [std::nested_exception::rethrow_nested\(\)](#).

throw_with_nested()

```
template<typename _Tp>
void std::throw_with_nested (
    _Tp && __t) [inline]
```

Throw an exception that also stores the currently active exception.

If `_Tp` is derived from `std::nested_exception` or is not usable as a base-class, throws a copy of `__t`. Otherwise, throws an object of an implementation-defined type derived from both `_Tp` and `std::nested_exception`, containing a copy of `__t` and the result of `std::current_exception()`.

In other words, throws the argument as a new exception that contains the currently active exception nested within it. This is intended for use in a catch handler to replace the caught exception with a different type, while still preserving the original exception. When the new exception is caught, the nested exception can be rethrown by using `std::rethrow_if_nested`.

This can be used at API boundaries, for example to catch a library's internal exception type and rethrow it nested with a `std::runtime_error`, or vice versa.

Since

C++11

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

uncaught_exception()

```
bool std::uncaught_exception () [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).

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

Referenced by [std::basic_ostream<_CharT, _Traits>::sentry::~~sentry\(\)](#), and [uncaught_exception\(\)](#).

3.6 Extensions

Collaboration diagram for Extensions:



Topics

- [Dynamic Bitset.](#)
- [Policy-Based Data Structures](#)
- [SGI](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std::tr2](#)

Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >`

3.6.1 Detailed Description

Components generally useful that are not part of any standard.

3.6.2 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



Classes

- struct `std::tr2::__dynamic_bitset_base<_WordT, _Alloc >`
- class `std::tr2::dynamic_bitset<_WordT, _Alloc >`

Functions

- template<typename _CharT, typename _Traits, typename _Alloc1>
void `std::tr2::dynamic_bitset<_WordT, _Alloc >::M_copy_to_string` (`std::basic_string<_CharT, _Traits, ↵_Alloc1 > &__str`, `_CharT __zero=_CharT('0')`, `_CharT __one=_CharT('1')`) const
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
`std::basic_ostream<_CharT, _Traits > & std::tr2::operator<<` (`std::basic_ostream<_CharT, _Traits > &__os`, const `dynamic_bitset<_WordT, _Alloc > &__x`)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
`std::basic_istream<_CharT, _Traits > & std::tr2::operator>>` (`std::basic_istream<_CharT, _Traits > &__↵is`, `dynamic_bitset<_WordT, _Alloc > &__x`)
- template<typename _WordT, typename _Alloc>
bool `std::tr2::operator!=` (const `dynamic_bitset<_WordT, _Alloc > &__lhs`, const `dynamic_bitset<_WordT, _Alloc > &__rhs`)
- template<typename _WordT, typename _Alloc>
bool `std::tr2::operator<=` (const `dynamic_bitset<_WordT, _Alloc > &__lhs`, const `dynamic_bitset<_WordT, _↵Alloc > &__rhs`)

- `template<typename _WordT, typename _Alloc>`
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

3.6.2.1 Detailed Description

3.6.2.2 Function Documentation

operator"!="()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator!= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator&()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
------------------	-----------

<code>__x</code>	A bitset of the same size as <code>__x</code> .
<code>__y</code>	

Returns

A new bitset.

These should be self-explanatory.

operator-()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

operator<<()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const dynamic_bitset< _WordT, _Alloc > & __x) [inline]
```

Stream output operator for `dynamic_bitset`.

operator<=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator<= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator> (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator>= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>>()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    dynamic_bitset< _WordT, _Alloc > & __x)
```

Stream input operator for dynamic_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

References [std::basic_string< _CharT, _Traits, _Alloc >::empty\(\)](#), [std::basic_string< _CharT, _Traits, _Alloc >::push_back\(\)](#), [std::basic_ios< _CharT, _Traits >::rdbuf\(\)](#), [std::basic_string< _CharT, _Traits, _Alloc >::reserve\(\)](#), [std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset\(\)](#), [std::basic_ios< _CharT, _Traits >::setstate\(\)](#), [std::basic_string< _CharT, _Traits, _Alloc >::size\(\)](#), [std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset\(\)](#) and [std::basic_ios< _CharT, _Traits >::widen\(\)](#).

operator^()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

Parameters

$_x$	A bitset.
$_y$	A bitset of the same size as $_x$.

Returns

A new bitset.

These should be self-explanatory.

operator" | ()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y) [inline]
```

Global bitwise operations on bitsets.

Parameters

$_x$	A bitset.
$_y$	A bitset of the same size as $_x$.

Returns

A new bitset.

These should be self-explanatory.

3.6.3 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Topics

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

Classes

- `struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >`

3.6.3.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

3.6.3.2 Containers

Collaboration diagram for Containers:



Topics

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

3.6.3.2.1 Detailed Description

3.6.3.2.2 Branch-Based

Collaboration diagram for Branch-Based:



Topics

- [Base and Policy Classes](#)

Classes

- class [__gnu_pbds::basic_branch](#)< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class [__gnu_pbds::tree](#)< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class [__gnu_pbds::trie](#)< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class [__gnu_pbds::detail::ov_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class [__gnu_pbds::detail::pat_trie_map](#)< Key, Mapped, Node_And_It_Traits, _Alloc >
- class [__gnu_pbds::detail::rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class [__gnu_pbds::detail::splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Detailed Description

3.6.3.2.3 Hash-Based

Collaboration diagram for Hash-Based:



Topics

- [Base and Policy Classes](#)

Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Ti, _Alloc >`
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`
- class `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

Detailed Description

3.6.3.2.4 Heap-Based

Collaboration diagram for Heap-Based:



Topics

- [Base and Policy Classes](#)

Classes

- class [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>](#)

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class [__gnu_pbds::detail::binary_heap<Value_Type, Cmp_Fn, _Alloc>](#)
- class [__gnu_pbds::detail::binomial_heap<Value_Type, Cmp_Fn, _Alloc>](#)
- class [__gnu_pbds::detail::pairing_heap<Value_Type, Cmp_Fn, _Alloc>](#)
- class [__gnu_pbds::detail::rc_binomial_heap<Value_Type, Cmp_Fn, _Alloc>](#)
- class [__gnu_pbds::detail::thin_heap<Value_Type, Cmp_Fn, _Alloc>](#)

Detailed Description

3.6.3.2.5 List-Based

Collaboration diagram for List-Based:



Classes

- class [__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >](#)

Macros

- `#define PB_DS_LU_BASE`

Detailed Description

3.6.3.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct [__gnu_pbds::container_error](#)
- struct [__gnu_pbds::insert_error](#)
- struct [__gnu_pbds::join_error](#)
- struct [__gnu_pbds::resize_error](#)

Functions

- void `__gnu_pbds::__throw_container_error()`
- void `__gnu_pbds::__throw_insert_error()`
- void `__gnu_pbds::__throw_join_error()`
- void `__gnu_pbds::__throw_resize_error()`

3.6.3.3.1 Detailed Description

3.6.3.4 Tags

Collaboration diagram for Tags:



Topics

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

Classes

- struct `__gnu_pbds::trivial_iterator_tag`

Typedefs

- typedef void `__gnu_pbds::trivial_iterator_difference_type`

3.6.3.4.1 Detailed Description

3.6.3.4.2 Typedef Documentation

`trivial_iterator_difference_type`

```
typedef void __gnu_pbds::trivial_iterator_difference_type
```

Prohibit moving trivial iterators.

3.6.3.4.3 Data Structure Type

Collaboration diagram for Data Structure Type:



Classes

- struct [__gnu_pbds::associative_tag](#)
- struct [__gnu_pbds::basic_branch_tag](#)
- struct [__gnu_pbds::basic_hash_tag](#)
- struct [__gnu_pbds::binary_heap_tag](#)
- struct [__gnu_pbds::binomial_heap_tag](#)
- struct [__gnu_pbds::cc_hash_tag](#)
- struct [__gnu_pbds::container_tag](#)
- struct [__gnu_pbds::gp_hash_tag](#)
- struct [__gnu_pbds::list_update_tag](#)
- struct [__gnu_pbds::ov_tree_tag](#)
- struct [__gnu_pbds::pairing_heap_tag](#)
- struct [__gnu_pbds::pat_trie_tag](#)
- struct [__gnu_pbds::priority_queue_tag](#)
- struct [__gnu_pbds::rb_tree_tag](#)
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
- struct [__gnu_pbds::sequence_tag](#)
- struct [__gnu_pbds::splay_tree_tag](#)
- struct [__gnu_pbds::string_tag](#)
- struct [__gnu_pbds::thin_heap_tag](#)
- struct [__gnu_pbds::tree_tag](#)
- struct [__gnu_pbds::trie_tag](#)

Detailed Description

3.6.3.4.4 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



Classes

- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::range_invalidation_guarantee](#)

Detailed Description

3.6.3.5 Traits

Collaboration diagram for Traits:



Classes

- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >](#)
- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >](#)
- struct [__gnu_pbds::container_traits< Cntr >](#)
- struct [__gnu_pbds::container_traits_base< _Tag >](#)
- struct [__gnu_pbds::container_traits_base< binary_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< binomial_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< cc_hash_tag >](#)
- struct [__gnu_pbds::container_traits_base< gp_hash_tag >](#)
- struct [__gnu_pbds::container_traits_base< list_update_tag >](#)
- struct [__gnu_pbds::container_traits_base< ov_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< pairing_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< pat_trie_tag >](#)
- struct [__gnu_pbds::container_traits_base< rb_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< rc_binomial_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< splay_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< thin_heap_tag >](#)
- struct [__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >](#)
- struct [__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >](#)
- struct [__gnu_pbds::detail::no_throw_copies< Key, Mapped >](#)
- struct [__gnu_pbds::detail::no_throw_copies< Key, null_type >](#)
- struct [__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >](#)
- struct [__gnu_pbds::null_type](#)
- struct [__gnu_pbds::detail::rebind_traits< _Alloc, T >](#)
- struct [__gnu_pbds::detail::select_value_type< Key, Mapped >](#)

- struct `__gnu_pbds::detail::select_value_type< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, false >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`

Variables

- template<typename Key, typename _Alloc, bool Store_Hash>
`null_type __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type`

3.6.3.5.1 Detailed Description

3.6.4 SGI

Collaboration diagram for SGI:



Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
- struct `__gnu_cxx::constant_void_fun< _Result >`
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
- struct `__gnu_cxx::select2nd< _Pair >`
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`

Functions

- template<typename _Tp>
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- constexpr size_t `std::bitset< _Nb >::Find_first` () const noexcept
- constexpr size_t `std::bitset< _Nb >::Find_next` (size_t __prev) const noexcept
- template<class _Operation1, class _Operation2>
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1` (const _Operation1 &__fn1, const _←
_Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2` (const _Operation1 &__fn1,
const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result>
`constant_void_fun< _Result > __gnu_cxx::constant0` (const _Result &__val)
- template<class _Result>
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1` (const _Result &__val)
- template<class _Result>
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2` (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
`std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n` (_InputIterator __first, _Size __count, _Output←
Iterator __result)
- template<typename _InputIterator, typename _Distance>
void `__gnu_cxx::distance` (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<class _Tp>
_Tp `__gnu_cxx::identity_element` (std::multiplies< _Tp >)
- template<class _Tp>
_Tp `__gnu_cxx::identity_element` (std::plus< _Tp >)

- `template<typename _InputIterator1, typename _InputIterator2>`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer>`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos, int __val) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_reset (size_t __pos) noexcept`
- `constexpr bitset< _Nb > & std::bitset< _Nb >::Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool std::bitset< _Nb >::Unchecked_test (size_t __pos) const noexcept`

3.6.4.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function `compose2` takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

3.6.4.2 Function Documentation

`__median()` [1/2]

```
template<typename _Tp>
const _Tp & __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c)
```

Find the median of three values.

Parameters

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.

Returns

One of `a`, `b` or `c`.

If $\{1,m,n\}$ is some convolution of $\{a,b,c\}$ such that $1 \leq m \leq n$ then the value returned will be m . This is an SGI extension.

`__median()` [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
```

```
const _Tp & __c,
_Compare __comp)
```

Find the median of three values using a predicate for comparison.

Parameters

<code>__a</code>	A value.
<code>__b</code>	A value.
<code>__c</code>	A value.
<code>__comp</code>	A binary predicate.

Returns

One of `a`, `b` or `c`.

If $\{l, m, n\}$ is some convolution of $\{a, b, c\}$ such that `comp(l, m)` and `comp(m, n)` are both true then the value returned will be `m`. This is an SGI extension.

`_Find_first()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_first () const [inline], [constexpr], [noexcept]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`_Find_next`

`_Find_next()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_next (
    size_t __prev) const [inline], [constexpr], [noexcept]
```

Finds the index of the next "on" bit after `prev`.

Returns

The index of the next bit set, or `size()` if not found.

Parameters

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

See also

`_Find_first`

_Unchecked_flip()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_flip (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by [flip\(\)](#).

_Unchecked_reset()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by [reset\(\)](#).

_Unchecked_set() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
    size_t __pos) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by [set\(\)](#).

_Unchecked_set() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
    size_t __pos,
    int __val) [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_test()

```
template<size_t _Nb>
bool std::bitset< _Nb >::_Unchecked_test (
    size_t __pos) const [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by [operator\[\]\(\)](#), and [test\(\)](#).

compose1()

```
template<class _Operation1, class _Operation2>
unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2) [inline]
```

An [SGI extension](#) .

compose2()

```
template<class _Operation1, class _Operation2, class _Operation3>
binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2,
    const _Operation3 & __fn3) [inline]
```

An [SGI extension](#) .

constant0()

```
template<class _Result>
constant_void_fun< _Result > __gnu_cxx::constant0 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

constant1()

```
template<class _Result>
constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

constant2()

```
template<class _Result>
constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (
    const _Result & __val) [inline]
```

An [SGI extension](#) .

copy_n()

```
template<typename _InputIterator, typename _Size, typename _OutputIterator>
std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (
    _InputIterator __first,
    _Size __count,
    _OutputIterator __result) [inline]
```

Copies the range [first,first+count) into [result,result+count).

Parameters

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

Returns

A std::pair composed of first+count and result+count.

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

distance()

```
template<typename _InputIterator, typename _Distance>
void __gnu_cxx::distance (
    _InputIterator __first,
    _InputIterator __last,
    _Distance & __n) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

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

Referenced by [__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>::temporary_buffer\(\)](#), and [distance\(\)](#).

identity_element() [1/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
    std::multiplies< _Tp > ) [inline]
```

An SGI extension .

identity_element() [2/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
    std::plus< _Tp > ) [inline]
```

An SGI extension .

lexicographical_compare_3way()

```
template<typename _InputIterator1, typename _InputIterator2>
int __gnu_cxx::lexicographical_compare_3way (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2)
```

memcmp on steroids.

Parameters

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

Returns

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

power() [1/2]

```
template<typename _Tp, typename _Integer>
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

power() [2/2]

```
template<typename _Tp, typename _Integer, typename _MonoidOperation>
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n,
    _MonoidOperation __monoid_op) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

random_sample() [1/2]

```
template<typename _InputIterator, typename _RandomAccessIterator>
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

random_sample() [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last,
    _RandomNumberGenerator & __rand) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

random_sample_n() [1/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n)
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

References `std::distance()`, `std::min()`, and `random_sample_n()`.

Referenced by `random_sample_n()`.

random_sample_n() [2/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n,
    _RandomNumberGenerator & __rand)
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

References [std::distance\(\)](#), and [std::min\(\)](#).

uninitialized_copy_n()

```
template<typename _InputIter, typename _Size, typename _ForwardIter>
std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (
    _InputIter __first,
    _Size __count,
    _ForwardIter __result) [inline]
```

Copies the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	Length
<code>__result</code>	An output iterator.

Returns

`__result + (__first + __count)`

Like `copy()`, but does not require an initialized output range.

3.7 Filesystem**Files**

- file [filesystem](#)

Classes

- class [std::filesystem::directory_entry](#)
- class [std::filesystem::directory_iterator](#)
- class [std::filesystem::file_status](#)
- class [std::filesystem::filesystem_error](#)
- class [std::filesystem::path::iterator](#)
- class [std::filesystem::path](#)
- class [std::filesystem::recursive_directory_iterator](#)
- struct [std::filesystem::space_info](#)

Typedefs

- using `std::filesystem::file_time_type`

Enumerations

- enum class `std::filesystem::copy_options` : unsigned short { `none` , `skip_existing` , `overwrite_existing` , `update_existing` , `recursive` , `copy_symlinks` , `skip_symlinks` , `directories_only` , `create_symlinks` , `create_hard_links` }
- enum class `std::filesystem::directory_options` : unsigned char { `none` , `follow_directory_symlink` , `skip_permission_denied` }
- enum class `std::filesystem::file_type` : signed char { `none` , `not_found` , `regular` , `directory` , `symlink` , `block` , `character` , `fifo` , `socket` , `unknown` }
- enum class `std::filesystem::perm_options` : unsigned { `replace` , `add` , `remove` , `nofollow` }
- enum class `std::filesystem::perms` : unsigned { `none` , `owner_read` , `owner_write` , `owner_exec` , `owner_all` , `group_read` , `group_write` , `group_exec` , `group_all` , `others_read` , `others_write` , `others_exec` , `others_all` , `all` , `set_uid` , `set_gid` , `sticky_bit` , `mask` , `unknown` }

Functions

- `path std::filesystem::absolute` (const `path` &__p)
- `path std::filesystem::absolute` (const `path` &__p, `error_code` &__ec)
- `path & std::filesystem::path::assign` (string_type &&__source)
- `iterator std::filesystem::path::begin` () const noexcept
- `path std::filesystem::canonical` (const `path` &__p)
- `path std::filesystem::canonical` (const `path` &__p, `error_code` &__ec)
- `int std::filesystem::path::compare` (const string_type &__s) const noexcept
- `int std::filesystem::path::compare` (const value_type * __s) const noexcept
- `void std::filesystem::copy` (const `path` &__from, const `path` &__to)
- `void std::filesystem::copy` (const `path` &__from, const `path` &__to, `copy_options` __options)
- `void std::filesystem::copy` (const `path` &__from, const `path` &__to, `copy_options` __options, `error_code` &__ec)
- `void std::filesystem::copy` (const `path` &__from, const `path` &__to, `error_code` &__ec)
- `bool std::filesystem::copy_file` (const `path` &__from, const `path` &__to)
- `bool std::filesystem::copy_file` (const `path` &__from, const `path` &__to, `copy_options` __option)
- `bool std::filesystem::copy_file` (const `path` &__from, const `path` &__to, `copy_options` __option, `error_code` &__ec)
- `bool std::filesystem::copy_file` (const `path` &__from, const `path` &__to, `error_code` &__ec)
- `void std::filesystem::copy_symlink` (const `path` &__existing_symlink, const `path` &__new_symlink)
- `void std::filesystem::copy_symlink` (const `path` &__existing_symlink, const `path` &__new_symlink, `error_code` &__ec) noexcept
- `bool std::filesystem::create_directories` (const `path` &__p)
- `bool std::filesystem::create_directories` (const `path` &__p, `error_code` &__ec)
- `bool std::filesystem::create_directory` (const `path` &__p)
- `bool std::filesystem::create_directory` (const `path` &__p, const `path` &__attributes)
- `bool std::filesystem::create_directory` (const `path` &__p, const `path` &__attributes, `error_code` &__ec) noexcept
- `bool std::filesystem::create_directory` (const `path` &__p, `error_code` &__ec) noexcept
- `void std::filesystem::create_directory_symlink` (const `path` &__to, const `path` &__new_symlink)

- void **std::filesystem::create_directory_symlink** (const [path](#) &__to, const [path](#) &__new_symlink, [error_code](#) &__ec) noexcept
- void **std::filesystem::create_hard_link** (const [path](#) &__to, const [path](#) &__new_hard_link)
- void **std::filesystem::create_hard_link** (const [path](#) &__to, const [path](#) &__new_hard_link, [error_code](#) &__ec) noexcept
- void **std::filesystem::create_symlink** (const [path](#) &__to, const [path](#) &__new_symlink)
- void **std::filesystem::create_symlink** (const [path](#) &__to, const [path](#) &__new_symlink, [error_code](#) &__ec) noexcept
- [path](#) **std::filesystem::current_path** ()
- void **std::filesystem::current_path** (const [path](#) &__p)
- void **std::filesystem::current_path** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- [path](#) **std::filesystem::current_path** ([error_code](#) &__ec)
- [iterator](#) **std::filesystem::path::end** () const noexcept
- bool **std::filesystem::equivalent** (const [path](#) &__p1, const [path](#) &__p2)
- bool **std::filesystem::equivalent** (const [path](#) &__p1, const [path](#) &__p2, [error_code](#) &__ec) noexcept
- bool **std::filesystem::exists** (const [path](#) &__p)
- bool **std::filesystem::exists** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::exists** ([file_status](#)) noexcept
- [path](#) **std::filesystem::path::extension** () const
- [uintmax_t](#) **std::filesystem::file_size** (const [path](#) &)
- [uintmax_t](#) **std::filesystem::file_size** (const [path](#) &, [error_code](#) &) noexcept
- [path](#) **std::filesystem::path::filename** () const
- [std::string](#) **std::filesystem::path::generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
[std::basic_string](#)<_CharT, _Traits, _Allocator> **std::filesystem::path::generic_string** (const _Allocator &__a= _Allocator()) const
- [std::u16string](#) **std::filesystem::path::generic_u16string** () const
- [std::u32string](#) **std::filesystem::path::generic_u32string** () const
- [std::string](#) **std::filesystem::path::generic_u8string** () const
- [std::wstring](#) **std::filesystem::path::generic_wstring** () const
- [uintmax_t](#) **std::filesystem::hard_link_count** (const [path](#) &)
- [uintmax_t](#) **std::filesystem::hard_link_count** (const [path](#) &, [error_code](#) &) noexcept
- bool **std::filesystem::path::has_extension** () const noexcept
- bool **std::filesystem::path::has_stem** () const noexcept
- [size_t](#) **std::filesystem::hash_value** (const [path](#) &__p) noexcept
- bool **std::filesystem::path::is_absolute** () const noexcept
- bool **std::filesystem::is_block_file** (const [path](#) &__p)
- bool **std::filesystem::is_block_file** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::is_block_file** ([file_status](#) __s) noexcept
- bool **std::filesystem::is_character_file** (const [path](#) &__p)
- bool **std::filesystem::is_character_file** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::is_character_file** ([file_status](#) __s) noexcept
- bool **std::filesystem::is_directory** (const [path](#) &__p)
- bool **std::filesystem::is_directory** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::is_directory** ([file_status](#) __s) noexcept
- bool **std::filesystem::is_empty** (const [path](#) &__p)
- bool **std::filesystem::is_empty** (const [path](#) &__p, [error_code](#) &__ec)
- bool **std::filesystem::is_fifo** (const [path](#) &__p)
- bool **std::filesystem::is_fifo** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::is_fifo** ([file_status](#) __s) noexcept
- bool **std::filesystem::is_other** (const [path](#) &__p)

- `bool std::filesystem::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_other (file_status) noexcept`
- `bool std::filesystem::is_regular_file (const path &__p)`
- `bool std::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_regular_file (file_status) noexcept`
- `bool std::filesystem::is_socket (const path &__p)`
- `bool std::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_socket (file_status __s) noexcept`
- `bool std::filesystem::is_symlink (const path &__p)`
- `bool std::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::filesystem::is_symlink (file_status) noexcept`
- `file_time_type std::filesystem::last_write_time (const path &)`
- `file_time_type std::filesystem::last_write_time (const path &, error_code &) noexcept`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `path & std::filesystem::path::make_preferred ()`
- `copy_options & std::filesystem::operator+= (copy_options &__x, copy_options __y) noexcept`
- `reference std::filesystem::path::iterator::operator* () const noexcept`
- `iterator & std::filesystem::path::iterator::operator++ () noexcept`
- `template<typename _CharT>
__detail::_Path2< _CharT * > & std::filesystem::path::operator+= (_CharT __x)`
- `path & std::filesystem::path::operator+= (basic_string_view< value_type > __x)`
- `path & std::filesystem::path::operator+= (const string_type &__x)`
- `path & std::filesystem::path::operator+= (const value_type * __x)`
- `path & std::filesystem::path::operator+= (value_type __x)`
- `iterator & std::filesystem::path::iterator::operator-- () noexcept`
- `path & std::filesystem::path::operator= (path &&) noexcept`
- `path & std::filesystem::path::operator= (string_type && __source)`
- `constexpr copy_options std::filesystem::operator^ (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::filesystem::operator^= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::filesystem::operator| (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::filesystem::operator|= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::filesystem::operator~ (copy_options __x) noexcept`
- `void std::filesystem::permissions (const path &, perms, perm_options, error_code &) noexcept`
- `void std::filesystem::permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `void std::filesystem::permissions (const path &__p, perms __prms, perm_options __opts=perm_options::↔ replace)`
- `path std::filesystem::proximate (const path &__p, const path &__base, error_code &__ec)`
- `path std::filesystem::proximate (const path &__p, const path &__base=current_path())`
- `path std::filesystem::proximate (const path &__p, error_code &__ec)`
- `path std::filesystem::read_symlink (const path &__p)`
- `path std::filesystem::read_symlink (const path &__p, error_code &__ec)`
- `path std::filesystem::relative (const path &__p, const path &__base, error_code &__ec)`
- `path std::filesystem::relative (const path &__p, const path &__base=current_path())`
- `path std::filesystem::relative (const path &__p, error_code &__ec)`
- `bool std::filesystem::remove (const path &, error_code &) noexcept`
- `bool std::filesystem::remove (const path &__p)`
- `uintmax_t std::filesystem::remove_all (const path &)`
- `uintmax_t std::filesystem::remove_all (const path &, error_code &)`
- `void std::filesystem::rename (const path &__from, const path &__to)`

- void **std::filesystem::rename** (const [path](#) &__from, const [path](#) &__to, [error_code](#) &__ec) noexcept
- void **std::filesystem::resize_file** (const [path](#) &__p, uintmax_t __size)
- void **std::filesystem::resize_file** (const [path](#) &__p, uintmax_t __size, [error_code](#) &__ec) noexcept
- [space_info](#) **std::filesystem::space** (const [path](#) &__p)
- [space_info](#) **std::filesystem::space** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- [file_status](#) **std::filesystem::status** (const [path](#) &__p)
- [file_status](#) **std::filesystem::status** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- bool **std::filesystem::status_known** ([file_status](#)) noexcept
- [path](#) **std::filesystem::path::stem** () const
- [string](#) **std::filesystem::path::string** () const
- template<typename _CharT, typename _Traits, typename _Allocator>
[basic_string](#)<_CharT, _Traits, _Allocator> **std::filesystem::path::string** (const _Allocator &__a) const
- void **swap** ([path](#) &__lhs, [path](#) &__rhs) noexcept
- void **std::filesystem::path::swap** ([path](#) &__rhs) noexcept
- [file_status](#) **std::filesystem::symlink_status** (const [path](#) &__p)
- [file_status](#) **std::filesystem::symlink_status** (const [path](#) &__p, [error_code](#) &__ec) noexcept
- [path](#) **std::filesystem::temp_directory_path** ()
- [path](#) **std::filesystem::temp_directory_path** ([error_code](#) &__ec)
- [u16string](#) **std::filesystem::path::u16string** () const
- [u32string](#) **std::filesystem::path::u32string** () const
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::_value_type_<__is_char_or_char8_t<_InputIterator>>>
[path](#) **u8path** (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::_value_type_is_char_or_<__char8_t<_Source>>>
[path](#) **u8path** (const _Source &__source)
- [string](#) **std::filesystem::path::u8string** () const
- [path](#) **std::filesystem::weakly_canonical** (const [path](#) &__p)
- [path](#) **std::filesystem::weakly_canonical** (const [path](#) &__p, [error_code](#) &__ec)
- [wstring](#) **std::filesystem::path::wstring** () const
- constexpr [perms](#) **std::filesystem::operator|** ([perms](#) __x, [perms](#) __y) noexcept
- constexpr [perms](#) **std::filesystem::operator^** ([perms](#) __x, [perms](#) __y) noexcept
- constexpr [perms](#) **std::filesystem::operator~** ([perms](#) __x) noexcept
- [perms](#) & **std::filesystem::operator&=** ([perms](#) &__x, [perms](#) __y) noexcept
- [perms](#) & **std::filesystem::operator|=** ([perms](#) &__x, [perms](#) __y) noexcept
- [perms](#) & **std::filesystem::operator^=** ([perms](#) &__x, [perms](#) __y) noexcept
- constexpr [perm_options](#) **std::filesystem::operator|** ([perm_options](#) __x, [perm_options](#) __y) noexcept
- constexpr [perm_options](#) **std::filesystem::operator^** ([perm_options](#) __x, [perm_options](#) __y) noexcept
- constexpr [perm_options](#) **std::filesystem::operator~** ([perm_options](#) __x) noexcept
- [perm_options](#) & **std::filesystem::operator&=** ([perm_options](#) &__x, [perm_options](#) __y) noexcept
- [perm_options](#) & **std::filesystem::operator|=** ([perm_options](#) &__x, [perm_options](#) __y) noexcept
- [perm_options](#) & **std::filesystem::operator^=** ([perm_options](#) &__x, [perm_options](#) __y) noexcept
- constexpr [directory_options](#) **std::filesystem::operator|** ([directory_options](#) __x, [directory_options](#) __y) noexcept
- constexpr [directory_options](#) **std::filesystem::operator^** ([directory_options](#) __x, [directory_options](#) __y) noexcept
- constexpr [directory_options](#) **std::filesystem::operator~** ([directory_options](#) __x) noexcept
- [directory_options](#) & **std::filesystem::operator&=** ([directory_options](#) &__x, [directory_options](#) __y) noexcept
- [directory_options](#) & **std::filesystem::operator|=** ([directory_options](#) &__x, [directory_options](#) __y) noexcept
- [directory_options](#) & **std::filesystem::operator^=** ([directory_options](#) &__x, [directory_options](#) __y) noexcept
- [directory_iterator](#) **begin** ([directory_iterator](#) __iter) noexcept
- [directory_iterator](#) **end** ([directory_iterator](#)) noexcept
- [recursive_directory_iterator](#) **begin** ([recursive_directory_iterator](#) __iter) noexcept
- [recursive_directory_iterator](#) **end** ([recursive_directory_iterator](#)) noexcept

3.7.1 Detailed Description

3.7.2 Typedef Documentation

file_time_type

using `std::filesystem::file_time_type`

The type used for file timestamps.

3.7.3 Enumeration Type Documentation

copy_options

enum class `std::filesystem::copy_options` : unsigned short [strong]

Bitmask type controlling effects of `filesystem::copy`.

directory_options

enum class `std::filesystem::directory_options` : unsigned char [strong]

Bitmask type controlling directory iteration.

file_type

enum class `std::filesystem::file_type` : signed char [strong]

Enumerated type representing the type of a file.

perm_options

enum class `std::filesystem::perm_options` : unsigned [strong]

Bitmask type controlling changes to permissions.

perms

enum class `std::filesystem::perms` : unsigned [strong]

Bitmask type representing file access permissions.

3.7.4 Function Documentation

begin() [1/2]

```
directory_iterator begin (  
    directory_iterator __iter) [related]
```

Enable range-based for using `directory_iterator`.

Since

C++17

e.g. for (auto& entry : `std::filesystem::directory_iterator(".")`) ...

begin() [2/2]

```
recursive_directory_iterator begin (  
    recursive_directory_iterator __iter) [related]
```

Enable range-based for using `recursive_directory_iterator`.

Since

C++17

e.g. for (auto& entry : `recursive_directory_iterator(".")`) ...

end() [1/2]

```
directory_iterator end (
    directory_iterator ) [related]
```

Return a past-the-end `directory_iterator`.

end() [2/2]

```
recursive_directory_iterator end (
    recursive_directory_iterator ) [related]
```

Return a past-the-end `recursive_directory_iterator`.

u8path() [1/2]

```
template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename
_CharT = __detail::_value_type_is_char_or_char8_t<_InputIterator>>
```

```
path u8path (
    _InputIterator __first,
    _InputIterator __last) [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References [u8path\(\)](#).

Referenced by [u8path\(\)](#), and [u8path\(\)](#).

u8path() [2/2]

```
template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __↵
detail::_value_type_is_char_or_char8_t<_Source>>
```

```
path u8path (
    const _Source & __source) [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References [u8path\(\)](#).

3.8 I/O

Collaboration diagram for I/O:



Classes

- class `std::basic_filebuf< _CharT, _Traits >`
- class `std::basic_fstream< _CharT, _Traits >`
- class `std::basic_ifstream< _CharT, _Traits >`
- class `std::basic_ios< _CharT, _Traits >`
- class `std::basic_iostream< _CharT, _Traits >`
- class `std::basic_istream< _CharT, _Traits >`
- class `std::basic_istreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_ofstream< _CharT, _Traits >`
- class `std::basic_ostream< _CharT, _Traits >`
- class `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_streambuf< _CharT, _Traits >`
- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`
- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`
- class `std::ios_base`
- class `std::istreambuf_iterator< _CharT, _Traits >`
- class `std::ostreambuf_iterator< _CharT, _Traits >`
- class `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`
- class `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`

Typedefs

- typedef `basic_filebuf< char > std::filebuf`
- typedef `basic_fstream< char > std::fstream`
- typedef `basic_ifstream< char > std::ifstream`
- typedef `basic_ios< char > std::ios`
- typedef `basic_iostream< char > std::iostream`
- typedef `basic_istream< char > std::istream`
- typedef `basic_istreamstream< char > std::istreamstream`
- typedef `basic_ofstream< char > std::ofstream`
- typedef `basic_ostream< char > std::ostream`
- typedef `basic_ostreamstream< char > std::ostreamstream`
- typedef `basic_streambuf< char > std::streambuf`
- typedef `basic_stringbuf< char > std::stringbuf`
- typedef `basic_stringstream< char > std::stringstream`
- typedef `basic_filebuf< wchar_t > std::wfilebuf`
- typedef `basic_fstream< wchar_t > std::wfstream`
- typedef `basic_ifstream< wchar_t > std::wifstream`
- typedef `basic_ios< wchar_t > std::wios`
- typedef `basic_iostream< wchar_t > std::wiostream`
- typedef `basic_istream< wchar_t > std::wistream`
- typedef `basic_istreamstream< wchar_t > std::wistreamstream`
- typedef `basic_ofstream< wchar_t > std::wofstream`
- typedef `basic_ostream< wchar_t > std::wostream`
- typedef `basic_ostreamstream< wchar_t > std::wostreamstream`
- typedef `basic_streambuf< wchar_t > std::wstreambuf`
- typedef `basic_stringbuf< wchar_t > std::wstringbuf`
- typedef `basic_stringstream< wchar_t > std::wstringstream`

3.8.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.↵html#std.io.objects>

3.8.2 Typedef Documentation

filebuf

```
typedef basic_filebuf<char> std::filebuf
```

Class for `char` file buffers.

fstream

```
typedef basic_fstream<char> std::fstream
```

Class for `char` mixed input and output file streams.

ifstream

```
typedef basic_ifstream<char> std::ifstream
```

Class for `char` input file streams.

ios

```
typedef basic_ios<char> std::ios
```

Base class for `char` streams.

iostream

```
typedef basic_iostream<char> std::iostream
```

Base class for `char` mixed input and output streams.

istream

```
typedef basic_istream<char> std::istream
```

Base class for `char` input streams.

istreamstream

```
typedef basic_istreamstream<char> std::istreamstream
```

Class for `char` input memory streams.

ofstream

```
typedef basic_ofstream<char> std::ofstream
```

Class for `char` output file streams.

ostream

```
typedef basic_ostream<char> std::ostream
```

Base class for `char` output streams.

ostringstream

```
typedef basic_ostringstream<char> std::ostringstream
```

Class for `char` output memory streams.

streambuf

```
typedef basic_streambuf<char> std::streambuf
```

Base class for `char` buffers.

stringbuf

```
typedef basic_stringbuf<char> std::stringbuf
```

Class for `char` memory buffers.

stringstream

```
typedef basic_stringstream<char> std::stringstream
```

Class for `char` mixed input and output memory streams.

wfilebuf

```
typedef basic_filebuf<wchar_t> std::wfilebuf
```

Class for `wchar_t` file buffers.

wfstream

```
typedef basic_fstream<wchar_t> std::wfstream
```

Class for `wchar_t` mixed input and output file streams.

wifstream

```
typedef basic_ifstream<wchar_t> std::wifstream
```

Class for `wchar_t` input file streams.

wios

```
typedef basic_ios<wchar_t> std::wios
```

Base class for `wchar_t` streams.

wiostream

```
typedef basic_iostream<wchar_t> std::wiostream
```

Base class for `wchar_t` mixed input and output streams.

wistream

```
typedef basic_istream<wchar_t> std::wistream
```

Base class for `wchar_t` input streams.

wistringstream

typedef `basic_istream<wchar_t>` `std::wistream`
 Class for `wchar_t` input memory streams.

wofstream

typedef `basic_ofstream<wchar_t>` `std::wofstream`
 Class for `wchar_t` output file streams.

wostream

typedef `basic_ostream<wchar_t>` `std::wostream`
 Base class for `wchar_t` output streams.

wostringstream

typedef `basic_ostream<wchar_t>` `std::wostringstream`
 Class for `wchar_t` output memory streams.

wstreambuf

typedef `basic_streambuf<wchar_t>` `std::wstreambuf`
 Base class for `wchar_t` buffers.

wstringbuf

typedef `basic_stringbuf<wchar_t>` `std::wstringbuf`
 Class for `wchar_t` memory buffers.

wstringstream

typedef `basic_stringstream<wchar_t>` `std::wstringstream`
 Class for `wchar_t` mixed input and output memory streams.

3.9 Iterators

Collaboration diagram for Iterators:



Topics

- [Iterator Tags](#)

Classes

- class [std::back_insert_iterator](#)< _Container >
- class [std::common_iterator](#)< _It, _Sent >
- class [std::counted_iterator](#)< _It >
- class [std::front_insert_iterator](#)< _Container >
- struct [std::input_iterator_tag](#)
- class [std::insert_iterator](#)< _Container >
- class [std::istream_iterator](#)< _Tp, _CharT, _Traits, _Dist >
- class [std::istreambuf_iterator](#)< _CharT, _Traits >
- struct [std::iterator](#)< _Category, _Tp, _Distance, _Pointer, _Reference >
- struct [std::iterator_traits](#)< _Iterator >
- struct [std::iterator_traits](#)< _Tp * >
- class [std::move_iterator](#)< _Iterator >
- class [std::move_sentinel](#)< _Sent >
- class [std::ostream_iterator](#)< _Tp, _CharT, _Traits >
- class [std::ostreambuf_iterator](#)< _CharT, _Traits >
- class [std::reverse_iterator](#)< _Iterator >

Macros

- `#define _GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

Typedefs

- `template<input_iterator _It>`
using **`std::const_iterator`**
- `template<semiregular _Sent>`
using **`std::const_sentinel`**
- `template<indirectly_readable _It>`
using **`std::iter_const_reference_t`**

Functions

- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::↵`
`copy_move_a2` (_CharT * __first, _CharT * __last, [ostreambuf_iterator](#)< _CharT > __result)
- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::↵`
`copy_move_a2` (const _CharT * __first, const _CharT * __last, [ostreambuf_iterator](#)< _CharT > __result)
- `template<bool _IsMove, typename _CharT>`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::↵`
`copy_move_a2` ([istreambuf_iterator](#)< _CharT > __first, [istreambuf_iterator](#)< _CharT > __last, _CharT * __result)
- `template<typename _CharT, typename _Size>`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::↵`
`copy_n_a` ([istreambuf_iterator](#)< _CharT > __it, _Size __n, _CharT * __result, bool __strict)
- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_↵`
`Iterator>::__value_type>::__value, _Iterator, move_iterator<_Iterator>>>>`
`constexpr _ReturnType std::↵`
`make_move_if_noexcept_iterator` (_Iterator __i)

- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>`
`constexpr _ReturnType std::make_move_if_noexcept_iterator (_Tp *__i)`
- `template<typename _Iterator>`
`constexpr reverse_iterator<_Iterator> std::make_reverse_iterator (_Iterator __i)`
- `template<typename _CharT, typename _Distance>`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, void>::__type std::advance (istreambuf_iterator<_CharT> &__i, _Distance __n)`
- `template<typename _Container>`
`constexpr back_insert_iterator<_Container> std::back_inserter (_Container &__x)`
- `template<typename _CharT>`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::copy (istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, ostreambuf_iterator<_CharT> __result)`
- `template<typename _CharT>`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, istreambuf_iterator<_CharT>>::__type std::find (istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, const _CharT &__val)`
- `template<typename _Container>`
`constexpr front_insert_iterator<_Container> std::front_inserter (_Container &__x)`
- `template<typename _Container>`
`constexpr insert_iterator<_Container> std::inserter (_Container &__x, std::__detail::__range_iter_t<_Container> __i)`
- `template<input_iterator _It>`
`constexpr const_iterator<_It> std::make_const_iterator (_It __it) noexcept(is_nothrow_convertible_v<_It, const_iterator<_It>>)`
- `template<semiregular _Sent>`
`constexpr const_sentinel<_Sent> std::make_const_sentinel (_Sent __s) noexcept(is_nothrow_convertible_v<_Sent, const_sentinel<_Sent>>)`
- `template<typename _Iterator>`
`constexpr move_iterator<_Iterator> std::make_move_iterator (_Iterator __i)`
- `template<typename _Iterator>`
`constexpr reverse_iterator<_Iterator> std::make_reverse_iterator (_Iterator __i)`
- `template<typename _IteratorL, typename _IteratorR>`
`requires requires { (__x.base() != __y.base()) -> convertible_to<bool>; }`
`constexpr bool std::operator!= (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y)`
- `template<typename _Iterator>`
`requires requires { (__x.base() + __n) -> same_as<_Iterator>; }`
`constexpr move_iterator<_Iterator> std::operator+ (typename move_iterator<_Iterator>::difference_type __n, const move_iterator<_Iterator> &__x)`
- `template<typename _Iterator>`
`constexpr reverse_iterator<_Iterator> std::operator+ (typename reverse_iterator<_Iterator>::difference_type __n, const reverse_iterator<_Iterator> &__x)`
- `template<typename _IteratorL, typename _IteratorR>`
`constexpr auto std::operator- (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y) -> decltype(__x.base() - __y.base())`
- `template<typename _IteratorL, typename _IteratorR>`
`constexpr auto std::operator- (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR>`
`requires requires { (__x.base() < __y.base()) -> convertible_to<bool>; }`
`constexpr bool std::operator< (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y)`

- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
constexpr bool std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
```
- ```
template<three_way_comparable _Iterator>
constexpr compare\_three\_way\_result\_t< _Iterator > std::operator<=> (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)
```
- ```
template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
constexpr compare_three_way_result_t< _IteratorL, _IteratorR > std::operator<=> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
```
- ```
template<three_way_comparable _Iterator>
constexpr compare\_three\_way\_result\_t< _Iterator, _Iterator > std::operator<=> (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)
```
- ```
template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
constexpr compare_three_way_result_t< _IteratorL, _IteratorR > std::operator<=> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
```
- ```
template<typename _CharT, typename _Traits>
bool std::operator== (const istreambuf\_iterator< _CharT, _Traits > &__a, const istreambuf\_iterator< _CharT, _Traits > &__b)
```
- ```
template<typename _Iterator>
constexpr bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator== (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)
```
- ```
template<typename _Iterator>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator== (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
constexpr bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)
```
- ```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
constexpr bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
```

- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR`  
`> &__y)`

## Variables

- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool std::disable_sized_sentinel_for< move\_iterator< _Iterator1 >, move\_iterator< _Iterator2`  
`> >`
- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool std::disable_sized_sentinel_for< reverse\_iterator< _Iterator1 >, reverse\_iterator< _↔`  
`Iterator2 > >`

### 3.9.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

### 3.9.2 Function Documentation

#### **back\_inserter()**

```
template<typename _Container>
back_insert_iterator< _Container > std::back_inserter (
 _Container & __x) [inline], [nodiscard], [constexpr]
```

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>_↔</code>  | A container of arbitrary type. |
| <code>__x</code> |                                |

#### Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you. Referenced by `std::match_results< \_BidirectionalIterator, polymorphic\_allocator< sub\_match< \_BidirectionalIterator > > >< const char * >>::format()`, `std::match_results< \_BidirectionalIterator, polymorphic\_allocator< sub\_match< \_BidirectionalIterator > > >< const char * >>::format()`, `regex_replace()`, `regex_replace()`, `regex_replace()`, and `regex_replace()`.

#### **front\_inserter()**

```
template<typename _Container>
front_insert_iterator< _Container > std::front_inserter (
 _Container & __x) [inline], [nodiscard], [constexpr]
```

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>_↔</code>  | A container of arbitrary type. |
| <code>__x</code> |                                |



**Returns**

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

**inserter()**

```
template<typename _Container>
insert_iterator< _Container > std::inserter (
 _Container & __x,
 std::__detail::__range_iter_t< _Container > __i) [nodiscard], [constexpr]
```

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__x</code> | A container of arbitrary type.  |
| <code>__i</code> | An iterator into the container. |

**Returns**

An instance of `insert_iterator` working on `__x`.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

**make\_reverse\_iterator()**

```
template<typename _Iterator>
reverse_iterator< _Iterator > std::make_reverse_iterator (
 _Iterator __i) [inline], [constexpr]
```

Generator function for `reverse_iterator`.

**operator==()**

```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
bool std::operator== (
 const reverse_iterator< _IteratorL > & __x,
 const reverse_iterator< _IteratorR > & __y) [nodiscard], [constexpr]
```

**Parameters**

|                  |                                   |
|------------------|-----------------------------------|
| <code>__x</code> | A <code>reverse_iterator</code> . |
| <code>__y</code> | A <code>reverse_iterator</code> . |

**Returns**

A simple bool.

Reverse iterators forward comparisons to their underlying base() iterators.

References [std::reverse\\_iterator<\\_Iterator>::base\(\)](#).

**3.9.3 Iterator Tags**

Collaboration diagram for Iterator Tags:

**Classes**

- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::contiguous\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

**3.9.3.1 Detailed Description**

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

**3.10 Locales****Classes**

- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar\\_t>](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)
- class [std::locale](#)
- class [std::messages<\\_CharT>](#)
- struct [std::messages\\_base](#)
- class [std::money\\_base](#)
- class [std::money\\_get<\\_CharT, \\_InIter>](#)
- class [std::money\\_put<\\_CharT, \\_OutIter>](#)
- class [std::moneypunct<\\_CharT, \\_Intl>](#)
- class [std::num\\_get<\\_CharT, \\_InIter>](#)

- class `std::num_put<_CharT, _OutIter >`
- class `std::num_punct<_CharT >`
- class `std::time_base`
- class `std::time_get<_CharT, _InIter >`
- class `std::time_put<_CharT, _OutIter >`
- class `std::wbuffer_convert<_Codecvt, _Elem, _Tr >`
- class `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`

## Functions

- template<typename \_Facet>  
bool `std::has_facet` (const `locale` &\_\_loc)
- template<typename \_Facet>  
const \_Facet & `std::use_facet` (const `locale` &\_\_loc)

### 3.10.1 Detailed Description

Classes and functions for internationalization and localization.

### 3.10.2 Function Documentation

#### `has_facet()`

```
template<typename _Facet>
bool std::has_facet (
 const locale & __loc) [inline], [nodiscard], [noexcept]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

#### Template Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

#### Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

#### Returns

true if `__loc` contains a facet of type `_Facet`, else false.

#### `use_facet()`

```
template<typename _Facet>
const _Facet & std::use_facet (
 const locale & __loc) [inline], [nodiscard]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

### Template Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

### Returns

Reference to facet of type `Facet`.

### Exceptions

|                                            |                                                                             |
|--------------------------------------------|-----------------------------------------------------------------------------|
| <a href="#"><code>std::bad_cast</code></a> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|--------------------------------------------|-----------------------------------------------------------------------------|

Referenced by [std::money\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_time\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::time\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [isalnum\(\)](#), [isalpha\(\)](#), [isblank\(\)](#), [iscntrl\(\)](#), [std::regex\\_traits<\\_Ch\\_type>::isctype\(\)](#), [isdigit\(\)](#), [isgraph\(\)](#), [islower\(\)](#), [isprint\(\)](#), [ispunct\(\)](#), [isspace\(\)](#), [isupper\(\)](#), [isxdigit\(\)](#), [std::regex\\_traits<\\_Ch\\_type>::lookup\\_classname\(\)](#), [std::regex\\_traits<\\_Ch\\_type>::lookup\\_collatename\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_iostream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), [std::time\\_put<\\_CharT, \\_OutIter>::put\(\)](#), [tolower\(\)](#), [toupper\(\)](#), [std::regex\\_traits<\\_CharT>::transform\(\)](#), [std::regex\\_traits<\\_CharT>::transform\\_primary\(\)](#), [std::regex\\_traits<\\_CharT>::translate\\_nocase\(\)](#), [std::regex\\_traits<\\_Ch\\_type>::value\(\)](#), and [ws\(\)](#).

### 3.11 Numerics

Collaboration diagram for Numerics:



#### Topics

- [Bit manipulation](#)
- [Complex Numbers](#)
- [Decimal Floating-Point Arithmetic](#)
- [Mathematical Special Functions](#)
- [Numeric Arrays](#)
- [Random Number Generation](#)
- [TR1 Mathematical Special Functions](#)

#### 3.11.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and mathematical special functions.

### 3.11.2 Bit manipulation

Collaboration diagram for Bit manipulation:



Utilities for examining and manipulating individual bits.

### 3.11.3 Complex Numbers

Collaboration diagram for Complex Numbers:



#### Classes

- class `std::complex<_Tp>`
- class `std::complex<double>`
- class `std::complex<float>`
- class `std::complex<long double>`

#### Functions

- `template<typename _Tp>`  
`_Tp std::__complex_abs` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`_Tp std::__complex_arg` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cos` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cosh` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_exp` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_log` (const `complex<_Tp>` &\_\_z)
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_pow` (const `complex<_Tp>` &\_\_x, const `complex<_Tp>` &\_\_y)

- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp constexpr std::norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &__t)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &__z)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &__z)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &__z)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &__t)`

- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &__z)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`complex< _Tp > &__x)`
- `template ostream & std::operator<< (ostream &, const complex< double > &)`
- `template ostream & std::operator<< (ostream &, const complex< float > &)`
- `template ostream & std::operator<< (ostream &, const complex< long double > &)`
- `template wostream & std::operator<< (wostream &, const complex< double > &)`
- `template wostream & std::operator<< (wostream &, const complex< float > &)`
- `template wostream & std::operator<< (wostream &, const complex< long double > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &__t)`
- `template<typename _Up>`  
`constexpr complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &__z)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp`  
`> &__x)`
- `template istream & std::operator>> (istream &, complex< double > &)`
- `template istream & std::operator>> (istream &, complex< float > &)`
- `template istream & std::operator>> (istream &, complex< long double > &)`
- `template wistream & std::operator>> (wistream &, complex< double > &)`
- `template wistream & std::operator>> (wistream &, complex< float > &)`
- `template wistream & std::operator>> (wistream &, complex< long double > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=__Tp(0))`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar (const _Tp &__rho,`  
`const _Up &__theta)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::pow (const complex< _Tp > &, int)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const _Tp &__x,`  
`const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const _Up &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const std::complex< _Up > &__y)`



- `template<typename _Tp>`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`

### 3.11.3.1 Detailed Description

Classes and functions for complex numbers.

### 3.11.3.2 Function Documentation

#### **abs()**

```
template<typename _Tp>
_Tp std::abs (
 const complex< _Tp > & __z) [inline]
```

Return magnitude of z.

Referenced by [abs\(\)](#), [fabs\(\)](#), [std::tr1::fabs\(\)](#), [std::binomial\\_distribution< \\_IntType >::operator\(\)\(\)](#), and [std::poisson\\_distribution< \\_IntType >](#)

#### **arg()**

```
template<typename _Tp>
_Tp std::arg (
 const complex< _Tp > & __z) [inline]
```

Return phase angle of z.

Referenced by [abs\(\)](#).

#### **conj()**

```
template<typename _Tp>
complex< _Tp > std::conj (
 const complex< _Tp > & __z) [inline], [constexpr]
```

Return complex conjugate of z.

#### **cos()**

```
template<typename _Tp>
complex< _Tp > std::cos (
 const complex< _Tp > & __z) [inline]
```

Return complex cosine of z.

Referenced by [polar\(\)](#).

#### **cosh()**

```
template<typename _Tp>
complex< _Tp > std::cosh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic cosine of z.

#### **exp()**

```
template<typename _Tp>
complex< _Tp > std::exp (
 const complex< _Tp > & __z) [inline]
```

Return complex base e exponential of z.

Referenced by [pow\(\)](#).

#### **fabs()**

```
template<typename _Tp>
std::complex< _Tp > std::tr1::fabs (
 const std::complex< _Tp > & __z) [inline]
```

[fabs\(\\_\\_z\)](#) [8.1.8].

References [std::abs\(\)](#).

**log()**

```
template<typename _Tp>
complex< _Tp > std::log (
 const complex< _Tp > & __z) [inline]
```

Return complex natural logarithm of z.

Referenced by [log10\(\)](#), [std::binomial\\_distribution< \\_IntType >::operator\(\)\(\)](#), [std::gamma\\_distribution< \\_RealType >::operator\(\)\(\)](#), [std::normal\\_distribution< \\_RealType >::operator\(\)\(\)](#), [std::poisson\\_distribution< \\_IntType >::operator\(\)\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [pow\(\)](#), and [pow\(\)](#).

**log10()**

```
template<typename _Tp>
complex< _Tp > std::log10 (
 const complex< _Tp > & __z) [inline]
```

Return complex base 10 logarithm of z.

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

**norm()**

```
template<typename _Tp>
_Tp constexpr std::norm (
 const complex< _Tp > & __z) [inline], [constexpr]
```

Return z magnitude squared.

**operator\*() [1/3]**

```
template<typename _Tp>
complex< _Tp > std::operator* (
 const _Tp & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x times y.

**operator\*() [2/3]**

```
template<typename _Tp>
complex< _Tp > std::operator* (
 const complex< _Tp > & __x,
 const _Tp & __y) [inline], [constexpr]
```

Return new complex value x times y.

**operator\*() [3/3]**

```
template<typename _Tp>
complex< _Tp > std::operator* (
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x times y.

**operator+() [1/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
 const _Tp & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value x plus y.

**operator+() [2/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
 const complex< _Tp > & __x) [inline], [constexpr]
Return x.
```

**operator+() [3/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
 const complex< _Tp > & __x,
 const _Tp & __y) [inline], [constexpr]
Return new complex value x plus y.
```

**operator+() [4/4]**

```
template<typename _Tp>
complex< _Tp > std::operator+ (
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
Return new complex value x plus y.
```

**operator-() [1/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
 const _Tp & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
Return new complex value x minus y.
```

**operator-() [2/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
 const complex< _Tp > & __x) [inline], [constexpr]
Return complex negation of x.
```

**operator-() [3/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
 const complex< _Tp > & __x,
 const _Tp & __y) [inline], [constexpr]
Return new complex value x minus y.
```

**operator-() [4/4]**

```
template<typename _Tp>
complex< _Tp > std::operator- (
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
Return new complex value x minus y.
```

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

```
template<typename _Tp>
complex< _Tp > std::operator/ (
 const _Tp & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value  $x$  divided by  $y$ .

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

```
template<typename _Tp>
complex< _Tp > std::operator/ (
 const complex< _Tp > & __x,
 const _Tp & __y) [inline], [constexpr]
```

Return new complex value  $x$  divided by  $y$ .

**operator/()** [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return new complex value  $x$  divided by  $y$ .

**operator<<()**

```
template<typename _Tp, typename _CharT, class _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const complex< _Tp > & __x)
```

Insertion operator for complex values.

References [std::ios\\_base::failbit](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::widen\(\)](#).

**operator==(())** [1/2]

```
template<typename _Tp>
bool std::operator==(
 const complex< _Tp > & __x,
 const _Tp & __y) [inline], [constexpr]
```

Return true if  $x$  is equal to  $y$ .

**operator==(())** [2/2]

```
template<typename _Tp>
bool std::operator==(
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline], [constexpr]
```

Return true if  $x$  is equal to  $y$ .

**operator>>()**

```
template<typename _Tp, typename _CharT, class _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __is,
 complex< _Tp > & __x)
```

Extraction operator for complex values.

### **polar()**

```
template<typename _Tp>
complex< _Tp > std::polar (
 const _Tp & __rho,
 const _Tp & __theta = _Tp(0)) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

References [cos\(\)](#), and [sin\(\)](#).

Referenced by [pow\(\)](#), and [pow\(\)](#).

### **pow()** [1/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
 const _Tp & __x,
 const complex< _Tp > & __y) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

References [log\(\)](#), [polar\(\)](#), and [pow\(\)](#).

### **pow()** [2/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
 const complex< _Tp > & __x,
 const _Tp & __y)
```

Return *x* to the *y*<sup>th</sup> power.

References [exp\(\)](#), [log\(\)](#), [polar\(\)](#), and [pow\(\)](#).

### **pow()** [3/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
 const complex< _Tp > & __x,
 const complex< _Tp > & __y) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

### **pow()** [4/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
 const complex< _Tp > & __z,
 int __n) [inline]
```

Return *x* to the *y*<sup>th</sup> power.

Referenced by [std::gamma\\_distribution< \\_RealType >::operator\(\)\(\)](#), [operator<<\(\)](#), [pow\(\)](#), [pow\(\)](#), and [std::tr1::pow\(\)](#).

### **pow()** [5/5]

```
template<typename _Tp, typename _Up>
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (
 const std::complex< _Tp > & __x,
 const _Up & __y) [inline]
```

Additional overloads [8.1.9].

References [std::pow\(\)](#).

**sin()**

```
template<typename _Tp>
complex< _Tp > std::sin (
 const complex< _Tp > & __z) [inline]
```

Return complex sine of *z*.

Referenced by [polar\(\)](#).

**sinh()**

```
template<typename _Tp>
complex< _Tp > std::sinh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic sine of *z*.

**sqrt()**

```
template<typename _Tp>
complex< _Tp > std::sqrt (
 const complex< _Tp > & __z) [inline]
```

Return complex square root of *z*.

Referenced by [std::normal\\_distribution<\\_RealType>::operator\(\)\(\)](#), and [std::student\\_t\\_distribution<\\_RealType>::operator\(\)\(\)](#).

**tan()**

```
template<typename _Tp>
complex< _Tp > std::tan (
 const complex< _Tp > & __z) [inline]
```

Return complex tangent of *z*.

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

**tanh()**

```
template<typename _Tp>
complex< _Tp > std::tanh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic tangent of *z*.

**3.11.4 Decimal Floating-Point Arithmetic**

Collaboration diagram for Decimal Floating-Point Arithmetic:



## Namespaces

- namespace [std::decimal](#)

### 3.11.4.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

### 3.11.5 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



## Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb>`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`



- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 ( _Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg ( _Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`
- `long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_if (float __nu, float __x)`
- `long double std::cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_jf (float __nu, float __x)`
- `long double std::cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_kf (float __nu, float __x)`
- `long double std::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_neumannf (float __nu, float __x)`
- `long double std::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_1f (float __k, float __phi)`
- `long double std::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_2f (float __k, float __phi)`
- `long double std::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float std::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::expint ( _Tp __x)`
- `float std::expintf (float __x)`
- `long double std::expintl (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)`
- `float std::hermitef (unsigned int __n, float __x)`
- `long double std::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)`
- `long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)`
- `float std::laguerref (unsigned int __n, float __x)`

- long double [std::laguerrel](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::legendre](#) (unsigned int \_\_l, \_Tp \_\_x)
- float [std::legendref](#) (unsigned int \_\_l, float \_\_x)
- long double [std::legendrel](#) (unsigned int \_\_l, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::riemann\\_zeta](#) (\_Tp \_\_s)
- float [std::riemann\\_zetaf](#) (float \_\_s)
- long double [std::riemann\\_zetal](#) (long double \_\_s)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_bessel](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_besself](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_bessell](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float [std::sph\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

### 3.11.5.1 Detailed Description

### 3.11.5.2 Mathematical Special Functions

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124 and then added to ISO C++ 2017.

#### 3.11.5.2.1 Introduction and History

The first significant library upgrade on the road to C++2011, [TR1](#), included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in `<cmath>`.

Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard [IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions](#).

For C++17 these functions were incorporated into the main standard.

#### 3.11.5.2.2 Contents

The following functions are implemented in namespace `std`:

- `assoc_laguerre` - Associated Laguerre functions
- `assoc_legendre` - Associated Legendre functions
- `beta` - Beta functions
- `comp_ellint_1` - Complete elliptic functions of the first kind
- `comp_ellint_2` - Complete elliptic functions of the second kind
- `comp_ellint_3` - Complete elliptic functions of the third kind
- `cyl_bessel_i` - Regular modified cylindrical Bessel functions
- `cyl_bessel_j` - Cylindrical Bessel functions of the first kind

- `cyl_bessel_k` - Irregular modified cylindrical Bessel functions
- `cyl_neumann` - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- `ellint_1` - Incomplete elliptic functions of the first kind
- `ellint_2` - Incomplete elliptic functions of the second kind
- `ellint_3` - Incomplete elliptic functions of the third kind
- `expint` - The exponential integral
- `hermite` - Hermite polynomials
- `laguerre` - Laguerre functions
- `legendre` - Legendre polynomials
- `riemann_zeta` - The Riemann zeta function
- `sph_bessel` - Spherical Bessel functions
- `sph_legendre` - Spherical Legendre functions
- `sph_neumann` - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- [conf\\_hyperg](#) - Confluent hypergeometric functions
- [hyperg](#) - Hypergeometric functions

#### 3.11.5.2.3 Argument Promotion

The arguments supplied to the non-suffixed functions will be promoted according to the following rules:

1. If any argument intended to be floating point is given an integral value That integral value is promoted to double.
2. All floating point arguments are promoted up to the largest floating point precision among them.

#### 3.11.5.2.4 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (`std::numeric_limits<Tp>::quiet_`↔NaN), the value NaN is returned.

#### 3.11.5.2.5 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with `std::numeric_limits` and functions thereof.

We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similarly, `long double` should give you more dynamic range and slightly more precision than `double` on many systems.

### 3.11.5.2.6 Testing

These functions have been tested against equivalent implementations from the [Gnu Scientific Library](#), [GSL](#) and [Boost](#) and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within  $10^{-15}$  for 64-bit double on linux-x86\_64 systems over most of the ranges of validity.

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

### 3.11.5.2.7 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55 Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example <http://people.math.sfu.ca/~cbm/aands/>

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: <http://dlmf.nist.gov/> This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

### 3.11.5.3 Function Documentation

#### airy\_ai()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (
 _Tp __x) [inline]
```

Return the Airy function  $Ai(x)$  of real argument x.

#### airy\_aif()

```
float __gnu_cxx::airy_aif (
 float __x) [inline]
```

Return the Airy function  $Ai(x)$  of float argument x.

#### airy\_ail()

```
long double __gnu_cxx::airy_ail (
 long double __x) [inline]
```

Return the Airy function  $Ai(x)$  of long double argument x.

#### airy\_bi()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (
 _Tp __x) [inline]
```

Return the Airy function  $Bi(x)$  of real argument x.

**airy\_bif()**

```
float __gnu_cxx::airy_bif (
 float __x) [inline]
```

Return the Airy function  $Bi(x)$  of float argument  $x$ .

**airy\_bil()**

```
long double __gnu_cxx::airy_bil (
 long double __x) [inline]
```

Return the Airy function  $Bi(x)$  of long double argument  $x$ .

**assoc\_laguerre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (
 unsigned int __n,
 unsigned int __m,
 _Tp __x) [inline]
```

Return the associated Laguerre polynomial of nonnegative order  $n$ , nonnegative degree  $m$  and real argument  $x$ :  $\leftarrow L_n^m(x)$ .

The associated Laguerre function of real degree  $\alpha$ ,  $L_n^\alpha(x)$ , is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!} {}_1F_1(-n; \alpha+1; x)$$

where  $(\alpha)_n$  is the Pochhammer symbol and  ${}_1F_1(a; c; x)$  is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree  $\alpha = m$  by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

and  $x \geq 0$ .

**See also**

laguerre for details of the Laguerre function of degree  $n$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |                                                                  |
|------------------|------------------------------------------------------------------|
| <code>__n</code> | The order of the Laguerre function, <code>__n</code> $\geq 0$ .  |
| <code>__m</code> | The degree of the Laguerre function, <code>__m</code> $\geq 0$ . |

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| <code>__x</code> | The argument of the Laguerre function, <code>__x &gt;= 0</code> . |
|------------------|-------------------------------------------------------------------|

### Exceptions

|                                |                              |
|--------------------------------|------------------------------|
| <code>std::domain_error</code> | if <code>__x &lt; 0</code> . |
|--------------------------------|------------------------------|

### assoc\_laguerref()

```
float std::assoc_laguerref (
 unsigned int __n,
 unsigned int __m,
 float __x) [inline]
```

Return the associated Laguerre polynomial of order `n`, degree `m`:  $L_n^m(x)$  for `float` argument.

#### See also

`assoc_laguerre` for more details.

### assoc\_laguerrel()

```
long double std::assoc_laguerrel (
 unsigned int __n,
 unsigned int __m,
 long double __x) [inline]
```

Return the associated Laguerre polynomial of order `n`, degree `m`:  $L_n^m(x)$ .

#### See also

`assoc_laguerre` for more details.

### assoc\_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (
 unsigned int __l,
 unsigned int __m,
 _Tp __x) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

The associated Legendre function is derived from the Legendre function  $P_l(x)$  by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

See also

`legendre` for details of the Legendre function of degree `l`

### Template Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__l</code> | The degree <code>__l &gt;= 0</code> .          |
| <code>__m</code> | The order <code>__m &lt;= l</code> .           |
| <code>__x</code> | The argument, <code>abs (__x) &lt;= 1</code> . |

### Exceptions

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__x) &gt; 1</code> . |
|--------------------------------|------------------------------------|

### `assoc_legendref()`

```
float std::assoc_legendref (
 unsigned int __l,
 unsigned int __m,
 float __x) [inline]
```

Return the associated Legendre function of degree `l` and order `m` for `float` argument.

See also

`assoc_legendre` for more details.

### `assoc_legendrel()`

```
long double std::assoc_legendrel (
 unsigned int __l,
 unsigned int __m,
 long double __x) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

See also

`assoc_legendre` for more details.

**beta()**

```
template<typename _Tpa, typename _Tpb>
__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (
 _Tpa __a,
 _Tpb __b) [inline]
```

Return the beta function,  $B(a, b)$ , for real parameters  $a, b$ .

The beta function is defined by

$$B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where  $a > 0$  and  $b > 0$

**Template Parameters**

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>_Tpa</code> | The floating-point type of the parameter <code>__a</code> . |
| <code>_Tpb</code> | The floating-point type of the parameter <code>__b</code> . |

**Parameters**

|                        |                                                                     |
|------------------------|---------------------------------------------------------------------|
| <code>↵<br/>__a</code> | The first argument of the beta function, <code>__a &gt; 0</code> .  |
| <code>↵<br/>__b</code> | The second argument of the beta function, <code>__b &gt; 0</code> . |

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::domain_error</code> | if <code>__a &lt; 0</code> or <code>__b &lt; 0</code> . |
|--------------------------------|---------------------------------------------------------|

**betaf()**

```
float std::betaf (
 float __a,
 float __b) [inline]
```

Return the beta function,  $B(a, b)$ , for `float` parameters  $a, b$ .

**See also**

`beta` for more details.

**betal()**

```
long double std::betal (
 long double __a,
 long double __b) [inline]
```

Return the beta function,  $B(a, b)$ , for long double parameters  $a, b$ .

**See also**

`beta` for more details.



**comp\_ellint\_1()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (
 _Tp __k) [inline]
```

Return the complete elliptic integral of the first kind  $K(k)$  for real modulus  $k$ .  
 The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

where  $F(k, \phi)$  is the incomplete elliptic integral of the first kind and the modulus  $|k| \leq 1$ .

See also

`ellint_1` for details of the incomplete elliptic function of the first kind.

**Template Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the modulus <code>__k</code> . |
|------------------|-----------------------------------------------------------|

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The modulus, <code>abs (__k) &lt;= 1</code> |
|------------------|---------------------------------------------|

**Exceptions**

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__k) &gt; 1</code> . |
|--------------------------------|------------------------------------|

**comp\_ellint\_1f()**

```
float std::comp_ellint_1f (
 float __k) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for `float` modulus  $k$ .

See also

`comp_ellint_1` for details.

**comp\_ellint\_1l()**

```
long double std::comp_ellint_1l (
 long double __k) [inline]
```

Return the complete elliptic integral of the first kind  $E(k)$  for long double modulus  $k$ .

See also

`comp_ellint_1` for details.

**comp\_ellint\_2()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (
 _Tp __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for real modulus  $k$ .

The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta$$

where  $E(k, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

**See also**

`ellint_2` for details of the incomplete elliptic function of the second kind.

**Template Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the modulus <code>__k</code> . |
|------------------|-----------------------------------------------------------|

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The modulus, <code>abs (__k) &lt;= 1</code> |
|------------------|---------------------------------------------|

**Exceptions**

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__k) &gt; 1</code> . |
|--------------------------------|------------------------------------|

**comp\_ellint\_2f()**

```
float std::comp_ellint_2f (
 float __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for `float` modulus  $k$ .

**See also**

`comp_ellint_2` for details.

**comp\_ellint\_2l()**

```
long double std::comp_ellint_2l (
 long double __k) [inline]
```

Return the complete elliptic integral of the second kind  $E(k)$  for long double modulus  $k$ .

**See also**

`comp_ellint_2` for details.

**comp\_ellint\_3()**

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (
 _Tp __k,
 _Tpn __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu) = \Pi(k, \nu, \pi/2)$  for real modulus  $k$ .  
The complete elliptic integral of the third kind is defined as

$$\Pi(k, \nu) = \Pi(k, \nu, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

where  $\Pi(k, \nu, \phi)$  is the incomplete elliptic integral of the second kind and the modulus  $|k| \leq 1$ .

See also

`ellint_3` for details of the incomplete elliptic function of the third kind.

**Template Parameters**

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>_Tp</code>  | The floating-point type of the modulus <code>__k</code> .   |
| <code>_Tpn</code> | The floating-point type of the argument <code>__nu</code> . |

**Parameters**

|                   |                                             |
|-------------------|---------------------------------------------|
| <code>__k</code>  | The modulus, <code>abs (__k) &lt;= 1</code> |
| <code>__nu</code> | The argument                                |

**Exceptions**

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__k) &gt; 1</code> . |
|--------------------------------|------------------------------------|

**comp\_ellint\_3f()**

```
float std::comp_ellint_3f (
 float __k,
 float __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `float` modulus  $k$ .

See also

`comp_ellint_3` for details.

**comp\_ellint\_3l()**

```
long double std::comp_ellint_3l (
 long double __k,
 long double __nu) [inline]
```

Return the complete elliptic integral of the third kind  $\Pi(k, \nu)$  for `long double` modulus  $k$ .

See also

`comp_ellint_3` for details.

### **conf\_hyperg()**

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
 _Tpa __a,
 _Tpc __c,
 _Tp __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

#### **Parameters**

|                       |                              |
|-----------------------|------------------------------|
| <code>↔<br/>_a</code> | The numeratorial parameter   |
| <code>↔<br/>_c</code> | The denominatorial parameter |
| <code>↔<br/>_x</code> | The argument                 |

### **conf\_hypergf()**

```
float __gnu_cxx::conf_hypergf (
 float __a,
 float __c,
 float __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

`conf_hyperg` for details.

### **conf\_hypergl()**

```
long double __gnu_cxx::conf_hypergl (
 long double __a,
 long double __c,
 long double __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

`conf_hyperg` for details.

**cyl\_bessel\_i()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (
 _Tpnu __nu,
 _Tp __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for real order  $\nu$  and argument  $x \geq 0$ .

The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

**Template Parameters**

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <code>_Tpnu</code> | The floating-point type of the order <code>__nu</code> .   |
| <code>_Tp</code>   | The floating-point type of the argument <code>__x</code> . |

**Parameters**

|                   |                                        |
|-------------------|----------------------------------------|
| <code>__nu</code> | The order                              |
| <code>__x</code>  | The argument, <code>__x &gt;= 0</code> |

**Exceptions**

|                                                |                              |
|------------------------------------------------|------------------------------|
| <a href="#"><code>std::domain_error</code></a> | if <code>__x &lt; 0</code> . |
|------------------------------------------------|------------------------------|

**cyl\_bessel\_if()**

```
float std::cyl_bessel_if (
 float __nu,
 float __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_i` for details.

**cyl\_bessel\_il()**

```
long double std::cyl_bessel_il (
 long double __nu,
 long double __x) [inline]
```

Return the regular modified Bessel function  $I_\nu(x)$  for `long double` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_i` for details.

**cyl\_bessel\_j()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (
 _Tpnu __nu,
 _Tp __x) [inline]
```

Return the Bessel function  $J_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

**Template Parameters**

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <code>_Tpnu</code> | The floating-point type of the order <code>__nu</code> .   |
| <code>_Tp</code>   | The floating-point type of the argument <code>__x</code> . |

**Parameters**

|                   |                                         |
|-------------------|-----------------------------------------|
| <code>__nu</code> | The order                               |
| <code>__x</code>  | The argument, <code>__x</code> $\geq 0$ |

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::domain_error</code></a> | if <code>__x</code> $< 0$ . |
|------------------------------------------------|-----------------------------|

**cyl\_bessel\_jf()**

```
float std::cyl_bessel_jf (
 float __nu,
 float __x) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for details.

**cyl\_bessel\_jl()**

```
long double std::cyl_bessel_jl (
 long double __nu,
 long double __x) [inline]
```

Return the Bessel function of the first kind  $J_\nu(x)$  for `long double` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_j` for details.

**cyl\_bessel\_k()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (
 _Tpnu __nu,
 _Tp __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  of real order  $\nu$  and argument  $x$ .

The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2} \frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}$$

where for integral  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ . For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

**Template Parameters**

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <code>_Tpnu</code> | The floating-point type of the order <code>__nu</code> .   |
| <code>_Tp</code>   | The floating-point type of the argument <code>__x</code> . |

**Parameters**

|                   |                                        |
|-------------------|----------------------------------------|
| <code>__nu</code> | The order                              |
| <code>__x</code>  | The argument, <code>__x &gt;= 0</code> |

**Exceptions**

|                                |                              |
|--------------------------------|------------------------------|
| <code>std::domain_error</code> | if <code>__x &lt; 0</code> . |
|--------------------------------|------------------------------|

**cyl\_bessel\_kf()**

```
float std::cyl_bessel_kf (
 float __nu,
 float __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for `float` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_k` for setails.

**cyl\_bessel\_kl()**

```
long double std::cyl_bessel_kl (
 long double __nu,
 long double __x) [inline]
```

Return the irregular modified Bessel function  $K_\nu(x)$  for `long double` order  $\nu$  and argument  $x \geq 0$ .

See also

`cyl_bessel_k` for setails.

### `cyl_neumann()`

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (
 _Tpnu __nu,
 _Tp __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of real order  $\nu$  and argument  $x \geq 0$ .

The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x) \cos \nu\pi - J_{-\nu}(x)}{\sin \nu\pi}$$

where  $x \geq 0$  and for integral order  $\nu = n$  a limit is taken:  $\lim_{\nu \rightarrow n}$ .

#### Template Parameters

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <code>_Tpnu</code> | The floating-point type of the order <code>__nu</code> .   |
| <code>_Tp</code>   | The floating-point type of the argument <code>__x</code> . |

#### Parameters

|                   |                                         |
|-------------------|-----------------------------------------|
| <code>__nu</code> | The order                               |
| <code>__x</code>  | The argument, <code>__x</code> $\geq 0$ |

#### Exceptions

|                                                |                              |
|------------------------------------------------|------------------------------|
| <a href="#"><code>std::domain_error</code></a> | if <code>__x &lt; 0</code> . |
|------------------------------------------------|------------------------------|

### `cyl_neumannf()`

```
float std::cyl_neumannf (
 float __nu,
 float __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `float` order  $\nu$  and argument  $x$ .

See also

`cyl_neumann` for setails.

### `cyl_neumannl()`

```
long double std::cyl_neumannl (
 long double __nu,
 long double __x) [inline]
```

Return the Neumann function  $N_\nu(x)$  of `long double` order  $\nu$  and argument  $x$ .



See also

`cyl_neumann` for setails.

### `ellint_1()`

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (
 _Tp __k,
 _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $F(k, \phi)$  for `real` modulus  $k$  and angle  $\phi$ . The incomplete elliptic integral of the first kind is defined as

$$F(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the first kind,  $K(k)$ .

See also

`comp_ellint_1`.

### Template Parameters

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>_Tp</code>  | The floating-point type of the modulus <code>__k</code> . |
| <code>_Tpp</code> | The floating-point type of the angle <code>__phi</code> . |

### Parameters

|                    |                                             |
|--------------------|---------------------------------------------|
| <code>__k</code>   | The modulus, <code>abs (__k) &lt;= 1</code> |
| <code>__phi</code> | The integral limit argument in radians      |

### Exceptions

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__k) &gt; 1</code> . |
|--------------------------------|------------------------------------|

### `ellint_1f()`

```
float std::ellint_1f (
 float __k,
 float __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for `float` modulus  $k$  and angle  $\phi$ .

See also

`ellint_1` for details.

**ellint\_1l()**

```
long double std::ellint_1l (
 long double __k,
 long double __phi) [inline]
```

Return the incomplete elliptic integral of the first kind  $E(k, \phi)$  for long double modulus  $k$  and angle  $\phi$ .

**See also**

`ellint_1` for details.

**ellint\_2()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (
 _Tp __k,
 _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^\phi \sqrt{1 - k^2 \sin^2 \theta}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the second kind,  $E(k)$ .

**See also**

`comp_ellint_2`.

**Template Parameters**

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>_Tp</code>  | The floating-point type of the modulus <code>__k</code> . |
| <code>_Tpp</code> | The floating-point type of the angle <code>__phi</code> . |

**Parameters**

|                    |                                             |
|--------------------|---------------------------------------------|
| <code>__k</code>   | The modulus, <code>abs (__k) &lt;= 1</code> |
| <code>__phi</code> | The integral limit argument in radians      |

**Returns**

The elliptic function of the second kind.

**Exceptions**

|                                |                                    |
|--------------------------------|------------------------------------|
| <code>std::domain_error</code> | if <code>abs (__k) &gt; 1</code> . |
|--------------------------------|------------------------------------|

**ellint\_2f()**

```
float std::ellint_2f (
 float __k,
 float __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$  for `float` argument.

See also

`ellint_2` for details.

**ellint\_2l()**

```
long double std::ellint_2l (
 long double __k,
 long double __phi) [inline]
```

Return the incomplete elliptic integral of the second kind  $E(k, \phi)$ .

See also

`ellint_2` for details.

**ellint\_3()**

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (
 _Tp __k,
 _Tpn __nu,
 _Tpp __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

For  $\phi = \pi/2$  this becomes the complete elliptic integral of the third kind,  $\Pi(k, \nu)$ .

See also

`comp_ellint_3`.

**Template Parameters**

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>_Tp</code>  | The floating-point type of the modulus <code>__k</code> .   |
| <code>_Tpn</code> | The floating-point type of the argument <code>__nu</code> . |
| <code>_Tpp</code> | The floating-point type of the angle <code>__phi</code> .   |

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__k</code> | The modulus, <code>abs(__k) &lt;= 1</code> |
|------------------|--------------------------------------------|

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__nu</code>  | The second argument                    |
| <code>__phi</code> | The integral limit argument in radians |

### Returns

The elliptic function of the third kind.

### Exceptions

|                                                |                                    |
|------------------------------------------------|------------------------------------|
| <a href="#"><code>std::domain_error</code></a> | if <code>abs (__k) &gt; 1</code> . |
|------------------------------------------------|------------------------------------|

### `ellint_3f()`

```
float std::ellint_3f (
 float __k,
 float __nu,
 float __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$  for `float` argument.

### See also

`ellint_3` for details.

### `ellint_3l()`

```
long double std::ellint_3l (
 long double __k,
 long double __nu,
 long double __phi) [inline]
```

Return the incomplete elliptic integral of the third kind  $\Pi(k, \nu, \phi)$ .

### See also

`ellint_3` for details.

### `expint()`

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::expint (
 _Tp __x) [inline]
```

Return the exponential integral  $Ei(x)$  for `real` argument `x`.

The exponential integral is given by

$$Ei(x) = - \int_{-x}^{\infty} \frac{e^t}{t} dt$$

### Template Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | The argument of the exponential integral function. |
|------------------|----------------------------------------------------|

**expintf()**

```
float std::expintf (
 float __x) [inline]
```

Return the exponential integral  $Ei(x)$  for `float` argument `x`.

**See also**

`expint` for details.

**expintl()**

```
long double std::expintl (
 long double __x) [inline]
```

Return the exponential integral  $Ei(x)$  for `long double` argument `x`.

**See also**

`expint` for details.

**hermite()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::hermite (
 unsigned int __n,
 _Tp __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of order `n` and `real` argument `x`.

The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |              |
|------------------|--------------|
| <code>__n</code> | The order    |
| <code>__x</code> | The argument |

**hermitef()**

```
float std::hermitef (
 unsigned int __n,
 float __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order n and float argument x.

**See also**

[hermite](#) for details.

**hermitel()**

```
long double std::hermitel (
 unsigned int __n,
 long double __x) [inline]
```

Return the Hermite polynomial  $H_n(x)$  of nonnegative order n and long double argument x.

**See also**

[hermite](#) for details.

**hyperg()**

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
 _Tpa __a,
 _Tpb __b,
 _Tpc __c,
 _Tp __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of real numeratorial parameters a and b, denominatorial parameter c, and argument x.

The hypergeometric function is defined by

$${}_2F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

**Parameters**

|                  |                                   |
|------------------|-----------------------------------|
| <code>__a</code> | The first numeratorial parameter  |
| <code>__b</code> | The second numeratorial parameter |
| <code>__c</code> | The denominatorial parameter      |
| <code>__x</code> | The argument                      |

**hypergf()**

```
float __gnu_cxx::hypergf (
 float __a,
 float __b,
 float __c,
 float __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**hypergl()**

```
long double __gnu_cxx::hypergl (
 long double __a,
 long double __b,
 long double __c,
 long double __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**laguerre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::laguerre (
 unsigned int __n,
 _Tp __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree n and real argument  $x \geq 0$ .

The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__n</code> | The nonnegative order                  |
| <code>__x</code> | The argument <code>__x</code> $\geq 0$ |

**Exceptions**

|                                |                              |
|--------------------------------|------------------------------|
| <code>std::domain_error</code> | if <code>__x &lt; 0</code> . |
|--------------------------------|------------------------------|

**laguerref()**

```
float std::laguerref (
 unsigned int __n,
 float __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree `n` and `float` argument  $x \geq 0$ .

**See also**

`laguerre` for more details.

**laguerrel()**

```
long double std::laguerrel (
 unsigned int __n,
 long double __x) [inline]
```

Returns the Laguerre polynomial  $L_n(x)$  of nonnegative degree `n` and `long double` argument  $x \geq 0$ .

**See also**

`laguerre` for more details.

**legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::legendre (
 unsigned int __l,
 _Tp __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and real argument  $|x| \leq 0$ .  
The Legendre function of order  $l$  and argument  $x$ ,  $P_l(x)$ , is defined by:

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2 - 1)^l$$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <code>__l</code> | The degree $l \geq 0$ |
|------------------|-----------------------|



|                  |                                             |
|------------------|---------------------------------------------|
| <code>__x</code> | The argument <code>abs (__x) &lt;= 1</code> |
|------------------|---------------------------------------------|

**Exceptions**

|                                |                                  |
|--------------------------------|----------------------------------|
| <code>std::domain_error</code> | if <code>abs (__x) &gt; 1</code> |
|--------------------------------|----------------------------------|

**legendref()**

```
float std::legendref (
 unsigned int __l,
 float __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and `float` argument  $|x| \leq 0$ .

**See also**

`legendre` for more details.

**legendrel()**

```
long double std::legendrel (
 unsigned int __l,
 long double __x) [inline]
```

Return the Legendre polynomial  $P_l(x)$  of nonnegative degree  $l$  and `long double` argument  $|x| \leq 0$ .

**See also**

`legendre` for more details.

**riemann\_zeta()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (
 _Tp __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for real argument  $s$ .

The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1 - 2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 \leq s \leq 1$$

For  $s < 1$  use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \Gamma(1-s) \zeta(1-s)$$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__s</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                         |                         |
|-------------------------|-------------------------|
| $\leftrightarrow$<br>_s | The argument $s \neq 1$ |
|-------------------------|-------------------------|

**riemann\_zetaf()**

```
float std::riemann_zetaf (
 float __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for float argument  $s$ .

**See also**

[riemann\\_zeta](#) for more details.

**riemann\_zetal()**

```
long double std::riemann_zetal (
 long double __s) [inline]
```

Return the Riemann zeta function  $\zeta(s)$  for long double argument  $s$ .

**See also**

[riemann\\_zeta](#) for more details.

**sph\_bessel()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (
 unsigned int __n,
 _Tp __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order  $n$  and real argument  $x \geq 0$ .  
The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

**Template Parameters**

|        |                                                   |
|--------|---------------------------------------------------|
| $\_Tp$ | The floating-point type of the argument $\_\_x$ . |
|--------|---------------------------------------------------|

**Parameters**

|                         |                               |
|-------------------------|-------------------------------|
| $\leftrightarrow$<br>_n | The integral order $n \geq 0$ |
| $\leftrightarrow$<br>_x | The real argument $x \geq 0$  |

**Exceptions**

|                                |                              |
|--------------------------------|------------------------------|
| <code>std::domain_error</code> | if <code>__x &lt; 0</code> . |
|--------------------------------|------------------------------|

**sph\_besself()**

```
float std::sph_besself (
 unsigned int __n,
 float __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order `n` and `float` argument  $x \geq 0$ .

See also

`sph_bessel` for more details.

**sph\_bessell()**

```
long double std::sph_bessell (
 unsigned int __n,
 long double __x) [inline]
```

Return the spherical Bessel function  $j_n(x)$  of nonnegative order `n` and `long double` argument  $x \geq 0$ .

See also

`sph_bessel` for more details.

**sph\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (
 unsigned int __l,
 unsigned int __m,
 _Tp __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and real angle  $\theta$  in radians. The spherical Legendre function is defined by

$$Y_l^m(\theta, \phi) = (-1)^m \left[ \frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!} \right] P_l^m(\cos \theta) \exp^{im\phi}$$

**Template Parameters**

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the angle <code>__theta</code> . |
|------------------|-------------------------------------------------------------|

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__l</code> | The order <code>__l</code> $\geq 0$ |
|------------------|-------------------------------------|

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| <code>__m</code>     | The degree <code>__m &gt;= 0</code> and <code>__m &lt;= __l</code> |
| <code>__theta</code> | The radian polar angle argument                                    |

**sph\_legendref()**

```
float std::sph_legendref (
 unsigned int __l,
 unsigned int __m,
 float __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and float angle  $\theta$  in radians.

See also

`sph_legendre` for details.

**sph\_legendrel()**

```
long double std::sph_legendrel (
 unsigned int __l,
 unsigned int __m,
 long double __theta) [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and long double angle  $\theta$  in radians.

See also

`sph_legendre` for details.

**sph\_neumann()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_neumann (
 unsigned int __n,
 _Tp __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and real argument  $x \geq 0$ .

The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

**Template Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>_Tp</code> | The floating-point type of the argument <code>__x</code> . |
|------------------|------------------------------------------------------------|

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__n</code> | The integral order $n \geq 0$ |
|------------------|-------------------------------|

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__x</code> | The real argument <code>__x &gt;= 0</code> |
|------------------|--------------------------------------------|

**Exceptions**

|                                |                              |
|--------------------------------|------------------------------|
| <code>std::domain_error</code> | if <code>__x &lt; 0</code> . |
|--------------------------------|------------------------------|

**sph\_neumannf()**

```
float std::sph_neumannf (
 unsigned int __n,
 float __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and `float` argument  $x \geq 0$ .

See also

`sph_neumann` for details.

**sph\_neumannl()**

```
long double std::sph_neumannl (
 unsigned int __n,
 long double __x) [inline]
```

Return the spherical Neumann function of integral order  $n \geq 0$  and `long double`  $x \geq 0$ .

See also

`sph_neumann` for details.

**3.11.6 Numeric Arrays**

Collaboration diagram for Numeric Arrays:

**Classes**

- class `std::gslice`
- class `std::gslice_array<_Tp>`
- class `std::indirect_array<_Tp>`
- class `std::mask_array<_Tp>`
- class `std::slice`
- class `std::slice_array<_Tp>`
- class `std::valarray<_Tp>`

## Functions

- [std::gslice::gslice](#) ()
- [std::gslice::gslice](#) (const [gslice](#) &)
- [std::gslice::gslice](#) (size\_t \_\_o, const [valarray](#)< size\_t > & \_\_l, const [valarray](#)< size\_t > & \_\_s)
- [std::gslice\\_array< \\_Tp >::gslice\\_array](#) (const [gslice\\_array](#) &)
- [std::indirect\\_array< \\_Tp >::indirect\\_array](#) (const [indirect\\_array](#) &)
- [std::mask\\_array< \\_Tp >::mask\\_array](#) (const [mask\\_array](#) &)
- [std::slice::slice](#) ()
- [std::slice::slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- [std::slice\\_array< \\_Tp >::slice\\_array](#) (const [slice\\_array](#) &)
- [std::valarray< \\_Tp >::valarray](#) () noexcept
- template<class \_Dom>
  - std::valarray< \_Tp >::valarray** (const [Expr](#)< \_Dom, \_Tp > & \_\_e)
- [std::valarray< \\_Tp >::valarray](#) (const \_Tp &, size\_t)
- template<typename \_Tp>
  - std::valarray< \_Tp >::valarray** (const \_Tp \*\_\_restrict \_\_p, size\_t \_\_n)
- [std::valarray< \\_Tp >::valarray](#) (const [gslice\\_array](#)< \_Tp > &)
- [std::valarray< \\_Tp >::valarray](#) (const [indirect\\_array](#)< \_Tp > &)
- [std::valarray< \\_Tp >::valarray](#) (const [mask\\_array](#)< \_Tp > &)
- [std::valarray< \\_Tp >::valarray](#) (const [slice\\_array](#)< \_Tp > &)
- [std::valarray< \\_Tp >::valarray](#) (const [valarray](#) &)
- [std::valarray< \\_Tp >::valarray](#) ([initializer\\_list](#)< \_Tp >)
- [std::valarray< \\_Tp >::valarray](#) (size\_t)
- [std::valarray< \\_Tp >::valarray](#) ([valarray](#) &&) noexcept
- [std::gslice::~gslice](#) ()
- [Expr](#)< \_ValFunClos< \_ValArray, \_Tp >, \_Tp > [std::valarray< \\_Tp >::apply](#) (\_Tp \_\_func(\_Tp)) const
- [Expr](#)< \_RefFunClos< \_ValArray, \_Tp >, \_Tp > [std::valarray< \\_Tp >::apply](#) (\_Tp \_\_func(const \_Tp &)) const
- [valarray](#)< \_Tp > [std::valarray< \\_Tp >::cshift](#) (int \_\_n) const
- \_Tp [std::valarray< \\_Tp >::max](#) () const
- \_Tp [std::valarray< \\_Tp >::min](#) () const
- [gslice](#) & [std::gslice::operator=](#) (const [gslice](#) &)
- template<class \_Dom>
  - void **std::gslice\_array< \_Tp >::operator=** (const [Expr](#)< \_Dom, \_Tp > &) const
- void [std::gslice\\_array< \\_Tp >::operator=](#) (const \_Tp &) const
- [gslice\\_array](#) & [std::gslice\\_array< \\_Tp >::operator=](#) (const [gslice\\_array](#) &)
- void [std::gslice\\_array< \\_Tp >::operator=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom>
  - void **std::indirect\_array< \_Tp >::operator=** (const [Expr](#)< \_Dom, \_Tp > &) const
- void [std::indirect\\_array< \\_Tp >::operator=](#) (const \_Tp &) const
- [indirect\\_array](#) & [std::indirect\\_array< \\_Tp >::operator=](#) (const [indirect\\_array](#) &)
- void [std::indirect\\_array< \\_Tp >::operator=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Ex>
  - void **std::mask\_array< \_Tp >::operator=** (const [Expr](#)< \_Ex, \_Tp > & \_\_e) const
- void [std::mask\\_array< \\_Tp >::operator=](#) (const \_Tp &) const
- [mask\\_array](#) & [std::mask\\_array< \\_Tp >::operator=](#) (const [mask\\_array](#) &)
- void [std::mask\\_array< \\_Tp >::operator=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom>
  - void **std::slice\_array< \_Tp >::operator=** (const [Expr](#)< \_Dom, \_Tp > &) const
- void [std::slice\\_array< \\_Tp >::operator=](#) (const \_Tp &) const
- [slice\\_array](#) & [std::slice\\_array< \\_Tp >::operator=](#) (const [slice\\_array](#) &)
- void [std::slice\\_array< \\_Tp >::operator=](#) (const [valarray](#)< \_Tp > &) const

```

• template<class _Dom>
 valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &__t)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const gslice_array<_Tp> &__ga)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> &__ia)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> &__ma)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const slice_array<_Tp> &__sa)
• valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &__v)
• valarray & std::valarray<_Tp>::operator= (initializer_list<_Tp> &__l)
• valarray<_Tp> & std::valarray<_Tp>::operator= (valarray<_Tp> &&__v) noexcept
• gslice_array<_Tp> std::valarray<_Tp>::operator[] (const gslice &__s)
• _Expr<_GClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[] (const gslice &__s) const
• mask_array<_Tp> std::valarray<_Tp>::operator[] (const valarray<bool> &__m)
• valarray<_Tp> std::valarray<_Tp>::operator[] (const valarray<bool> &__m) const
• indirect_array<_Tp> std::valarray<_Tp>::operator[] (const valarray<size_t> &__i)
• _Expr<_IClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[] (const valarray<size_t> &__i) const
• _Tp & std::valarray<_Tp>::operator[] (size_t __i) noexcept
• const _Tp & std::valarray<_Tp>::operator[] (size_t) const noexcept
• slice_array<_Tp> std::valarray<_Tp>::operator[] (slice __s)
• _Expr<_SClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::operator[] (slice __s) const
• void std::valarray<_Tp>::resize (size_t __size, _Tp __c=_Tp())
• valarray<_Tp> std::valarray<_Tp>::shift (int __n) const
• valarray<size_t> std::gslice::size () const
• size_t std::slice::size () const
• size_t std::valarray<_Tp>::size () const
• size_t std::gslice::start () const
• size_t std::slice::start () const
• valarray<size_t> std::gslice::stride () const
• size_t std::slice::stride () const
• _Tp std::valarray<_Tp>::sum () const
• void std::valarray<_Tp>::swap (valarray<_Tp> &__v) noexcept
• template<typename _Tp, size_t _Nm>
 std::valarray (const _Tp(&)[_Nm], size_t) -> valarray<_Tp>

```

### 3.11.6.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

### 3.11.6.2 Function Documentation

#### gslice() [1/3]

```
std::gslice::gslice () [inline]
```

Construct an empty slice.

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

Referenced by [gslice\(\)](#), [gslice\(\)](#), [gslice\(\)](#), and [operator=\(\)](#).

#### gslice() [2/3]

```
std::gslice::gslice (
 const gslice & __g) [inline]
```

Copy constructor.

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

**gslice()** [3/3]

```
std::gslice::gslice (
 size_t __o,
 const valarray< size_t > & __l,
 const valarray< size_t > & __s) [inline]
```

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

**Parameters**

|                                                                                                                    |                                                    |
|--------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| <a href="#"> <code>__o</code></a> | Offset in array of first element.                  |
| <a href="#"> <code>__l</code></a> | Array of dimension lengths.                        |
| <a href="#"> <code>__s</code></a> | Array of dimension strides between array elements. |

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

**gslice\_array()**

```
template<typename _Tp>
std::gslice_array< _Tp >::gslice_array (
 const gslice_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [gslice\\_array\(\)](#).

Referenced by [gslice\\_array\(\)](#), and [operator=\(\)](#).

**indirect\_array()**

```
template<typename _Tp>
std::indirect_array< _Tp >::indirect_array (
 const indirect_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [indirect\\_array\(\)](#).

Referenced by [indirect\\_array\(\)](#), and [operator=\(\)](#).

**mask\_array()**

```
template<typename _Tp>
std::mask_array< _Tp >::mask_array (
 const mask_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [mask\\_array\(\)](#).

Referenced by [mask\\_array\(\)](#), and [operator=\(\)](#).

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

```
std::slice::slice () [inline]
```

Construct an empty slice.



Referenced by [operator==](#).

### slice() [2/2]

```
std::slice::slice (
 size_t __o,
 size_t __d,
 size_t __s) [inline]
```

Construct a slice.

#### Parameters

|                                                                                                                       |                                   |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"></a><br><code>__o</code> | Offset in array of first element. |
| <a href="#"></a><br><code>__d</code> | Number of elements in slice.      |
| <a href="#"></a><br><code>__s</code> | Stride between array elements.    |

### slice\_array()

```
template<typename _Tp>
std::slice_array< _Tp >::slice_array (
 const slice_array< _Tp > & __a) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

References [slice\\_array\(\)](#).

Referenced by [slice\\_array\(\)](#), and [operator=\(\)](#).

### valarray() [1/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray () [inline], [noexcept]
```

Construct an empty array.

Referenced by [valarray\(\)](#), [valarray\(\)](#), [valarray\(\)](#), [cshift\(\)](#), [operator%=\(, operator%=\(, operator&=\(, operator&=\(, operator\\*=\(, operator\\*=\(, operator+=\(, operator+=\(, operator-=\(, operator-=\(, operator/=\(, operator/=\(, operator<<=\(, operator=\(, operator=\(, operator>=\(\), operator>=\(\), operator\[ \]\(\), operator\[ \]\(\), operator\[ \]\(\), operator\[ \]\(\), operator^=\(, operator^=\(, operator|=\(, shift\(\), and swap\(\)](#).

### valarray() [2/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const _Tp & __t,
 size_t __n) [inline]
```

Construct an array with  $n$  elements initialized to  $t$ .

### valarray() [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const gslice_array< _Tp > & __ga) [inline]
```

Construct an array with the same size and values in  $ga$ .

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

**valarray()** [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const indirect_array< _Tp > & __ia) [inline]
```

Construct an array with the same size and values in *ia*.

**valarray()** [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const mask_array< _Tp > & __ma) [inline]
```

Construct an array with the same size and values in *ma*.

**valarray()** [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const slice_array< _Tp > & __sa) [inline]
```

Construct an array with the same size and values in *sa*.

**valarray()** [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 const valarray< _Tp > & __v) [inline]
```

Copy constructor.

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

**valarray()** [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 initializer_list< _Tp > __l) [inline]
```

Construct an array with an `initializer_list` of values.

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

**valarray()** [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 size_t __n) [inline], [explicit]
```

Construct an array with *n* elements.

**valarray()** [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
 valarray< _Tp > && __v) [inline], [noexcept]
```

Move constructor.

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

**~gslice()**

```
std::gslice::~gslice () [inline]
```

Destructor.

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

```
template<class _Tp>
_Expr< _ValFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
 _Tp __func_Tp) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying \_\_func to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

|                     |                                       |
|---------------------|---------------------------------------|
| <code>__func</code> | Function of Tp returning Tp to apply. |
|---------------------|---------------------------------------|

**Returns**

New valarray with transformed elements.

**apply()** [2/2]

```
template<class _Tp>
_Expr< _RefFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
 _Tp __funcconst _Tp &) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying \_\_func to the corresponding element of this array. The new array has the same size as this one.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__func</code> | Function of const Tp& returning Tp to apply. |
|---------------------|----------------------------------------------|

**Returns**

New valarray with transformed elements.

**cshift()**

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::cshift (
 int __n) const [inline]
```

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is  $(i - n) \% \text{size}()$ . The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__n</code> | Number of element positions to rotate. |
|------------------|----------------------------------------|

**Returns**

New valarray with elements in shifted positions.

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

**max()**

```
template<typename _Tp>
_Tp std::valarray<_Tp>::max () const [inline]
Return the maximum element using operator<().
```

**min()**

```
template<typename _Tp>
_Tp std::valarray<_Tp>::min () const [inline]
Return the minimum element using operator<().
```

**operator=()** [1/20]

```
gslice & std::gslice::operator= (
 const gslice & __g) [inline]
```

Assignment operator.

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

**operator=()** [2/20]

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator= (
 const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [3/20]

```
template<typename _Tp>
gslice_array<_Tp> & std::gslice_array<_Tp>::operator= (
 const gslice_array<_Tp> & __a) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

References [gslice\\_array\(\)](#).

**operator=()** [4/20]

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator= (
 const valarray<_Tp> & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

References [std::valarray<\\_Tp>::size\(\)](#).

**operator=()** [5/20]

```
template<typename _Tp>
void std::indirect_array<_Tp>::operator= (
 const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [6/20]

```
template<typename _Tp>
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
 const indirect_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

References [indirect\\_array\(\)](#).

**operator=()** [7/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
 const valarray< _Tp > & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

**operator=()** [8/20]

```
template<typename _Tp>
void std::mask_array< _Tp >::operator= (
 const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [9/20]

```
template<typename _Tp>
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
 const mask_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

References [mask\\_array\(\)](#).

**operator=()** [10/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
 const _Tp & __t) const [inline]
```

Assign all slice elements to *t*.

**operator=()** [11/20]

```
template<typename _Tp>
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
 const slice_array< _Tp > & __a) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

References [slice\\_array\(\)](#).

**operator=()** [12/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
 const valarray< _Tp > & __v) const [inline]
```

Assign slice elements to corresponding elements of *v*.

**operator=()** [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const _Tp & __t) [inline]
```

Assign elements to a value.

Assign all elements of array to *t*.

**Parameters**

|          |                     |
|----------|---------------------|
| ↔        | Value for elements. |
| ↔        |                     |
| ↔        |                     |
| ↔        |                     |
| <i>t</i> |                     |

**operator=()** [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const gslice_array< _Tp > & __ga) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

**Parameters**

|      |                                 |
|------|---------------------------------|
| __ga | Array slice to get values from. |
|------|---------------------------------|

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

**operator=()** [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const indirect_array< _Tp > & __ia) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

**Parameters**

|           |                                 |
|-----------|---------------------------------|
| ↔         | Array slice to get values from. |
| <i>ia</i> |                                 |

**operator=()** [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const mask_array< _Tp > & __ma) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__ma</code> | Array slice to get values from. |
|-------------------|---------------------------------|

**operator=()** [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const slice_array< _Tp > & __sa) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in `sa`. Results are undefined if `sa` does not have the same size as this array.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__sa</code> | Array slice to get values from. |
|-------------------|---------------------------------|

**operator=()** [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 const valarray< _Tp > & __v) [inline]
```

Assign elements to an array.

Assign elements of array to values in `v`.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Valarray to get values from. |
|------------------|------------------------------|

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

**operator=()** [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
 initializer_list< _Tp > __l) [inline]
```

Assign elements to an `initializer_list`.

Assign elements of array to values in `__l`. Results are undefined if `__l` does not have the same size as this array.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__l</code> | initializer_list to get values from. |
|------------------|--------------------------------------|

**operator=()** [20/20]

```
template<typename _Tp>
```

```
valarray< _Tp > & std::valarray< _Tp >::operator= (
 valarray< _Tp > && __v) [inline], [noexcept]
```

Move assign elements to an array.

Move assign elements of array to values in *v*.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Valarray to get values from. |
|------------------|------------------------------|

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

#### **operator[]()** [1/9]

```
template<typename _Tp>
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
 const gslice & __s) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the *gslice* argument. The new valarray has the same size as the input *gslice*.

#### See also

[gslice](#).

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | The source <i>gslice</i> . |
|------------------|----------------------------|

#### Returns

New valarray containing elements in *\_\_s*.

#### **operator[]()** [2/9]

```
template<typename _Tp>
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
 const gslice & __s) const [inline]
```

Return an array subset.

Returns a *slice\_array* referencing the elements of the array indicated by the *slice* argument.

#### See also

[gslice](#).

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__s</code> | The source slice. |
|------------------|-------------------|



**Returns**

Slice\_array referencing elements indicated by \_\_s.

**operator[]()** [3/9]

```
template<typename _Tp>
mask_array< _Tp > std::valarray< _Tp >::operator[] (
 const valarray< bool > & __m) [inline]
```

Return a reference to an array subset.

Returns a new mask\_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

**Parameters**

|                          |                       |
|--------------------------|-----------------------|
| $\leftrightarrow$<br>__m | The valarray bitmask. |
|--------------------------|-----------------------|

**Returns**

New valarray containing elements indicated by \_\_m.

References [valarray\(\)](#), and [size\(\)](#).

**operator[]()** [4/9]

```
template<typename _Tp>
valarray< _Tp > std::valarray< _Tp >::operator[] (
 const valarray< bool > & __m) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

**Parameters**

|                          |                       |
|--------------------------|-----------------------|
| $\leftrightarrow$<br>__m | The valarray bitmask. |
|--------------------------|-----------------------|

**Returns**

New valarray containing elements indicated by \_\_m.

References [valarray\(\)](#), and [size\(\)](#).

**operator[]()** [5/9]

```
template<typename _Tp>
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
 const valarray< size_t > & __i) [inline]
```

Return a reference to an array subset.

Returns an indirect\_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect\_array refers to these elements.

**Parameters**

|          |                                  |
|----------|----------------------------------|
| ↩        | The valarray element index list. |
| ↩        |                                  |
| ↩        |                                  |
| ↩        |                                  |
| <i>i</i> |                                  |

**Returns**

Indirect\_array referencing elements in `__i`.

References [valarray\(\)](#), and [size\(\)](#).

**operator[]() [6/9]**

```
template<typename _Tp>
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
 const valarray< size_t > & __i) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

**Parameters**

|          |                                  |
|----------|----------------------------------|
| ↩        | The valarray element index list. |
| ↩        |                                  |
| ↩        |                                  |
| ↩        |                                  |
| <i>i</i> |                                  |

**Returns**

New valarray containing elements in `__s`.

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

**operator[]() [7/9]**

```
template<typename _Tp>
_Tp & std::valarray< _Tp >::operator[] (
 size_t __i) [inline], [noexcept]
```

Return a reference to the *i*'th array element.

**Parameters**

|          |                             |
|----------|-----------------------------|
| ↩        | Index of element to return. |
| ↩        |                             |
| ↩        |                             |
| ↩        |                             |
| <i>i</i> |                             |

**Returns**

Reference to the i'th element.

**operator[]()** [8/9]

```
template<typename _Tp>
slice_array< _Tp > std::valarray< _Tp >::operator[] (
 slice __s) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

slice.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__↔</code> | The source slice. |
| <code>__s</code> |                   |

**Returns**

New valarray containing elements in \_\_s.

**operator[]()** [9/9]

```
template<typename _Tp>
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
 slice __s) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

**See also**

slice.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__↔</code> | The source slice. |
| <code>__s</code> |                   |

**Returns**

New valarray containing elements in \_\_s.

**resize()**

```
template<class _Tp>
void std::valarray< _Tp >::resize (
```

```
size_t __size,
_Tp __c = _Tp()) [inline]
```

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

#### Parameters

|                     |                             |
|---------------------|-----------------------------|
| <code>__size</code> | New array size.             |
| <code>__c</code>    | New value for all elements. |

#### shift()

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::shift (
 int __n) const [inline]
```

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | Number of element positions to shift. |
|------------------|---------------------------------------|

#### Returns

New valarray with elements in shifted positions.

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

#### size() [1/3]

```
valarray< size_t > std::gslice::size () const [inline]
```

Return array of sizes of slice dimensions.

#### size() [2/3]

```
size_t std::slice::size () const [inline]
```

Return size of slice.

#### size() [3/3]

```
template<class _Tp>
size_t std::valarray< _Tp >::size () const [inline]
```

Return the number of elements in array.

Referenced by [valarray\(\)](#), [valarray\(\)](#), [std::gslice\\_array< \\_Tp >::operator=\(\)](#), [operator=\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

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

```
size_t std::gslice::start () const [inline]
```

Return array offset of first slice element.

**start()** [2/2]

```
size_t std::slice::start () const [inline]
```

Return array offset of first slice element.

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

```
valarray< size_t > std::gslice::stride () const [inline]
```

Return array of array strides for each dimension.

**stride()** [2/2]

```
size_t std::slice::stride () const [inline]
```

Return array stride of slice.

**sum()**

```
template<class _Tp>
_Tp std::valarray< _Tp >::sum () const [inline]
```

Return the sum of all elements in the array.  
Accumulates the sum of all elements into a `Tp` using `+=`. The order of adding the elements is unspecified.

**swap()**

```
template<class _Tp>
void std::valarray< _Tp >::swap (
 valarray< _Tp > & __v) [inline], [noexcept]
```

Swap.  
References [valarray\(\)](#).

**3.11.7 Random Number Generation**

Collaboration diagram for Random Number Generation:

**Topics**

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

## Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>`  
`_RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`

### 3.11.7.1 Detailed Description

A facility for generating random numbers on selected distributions.

### 3.11.7.2 Function Documentation

#### `generate_canonical()`

```
template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
_RealType std::generate_canonical (
 _UniformRandomNumberGenerator & __g)
```

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Referenced by [generate\\_canonical\(\)](#).

### 3.11.7.3 Random Number Distributions

Collaboration diagram for Random Number Distributions:



## Topics

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Sampling Distributions](#)
- [Uniform Distributions](#)

### 3.11.7.3.1 Detailed Description

### 3.11.7.3.2 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



## Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution<_IntType>`
- class `std::geometric_distribution<_IntType>`
- class `std::negative_binomial_distribution<_IntType>`

## Functions

- `template<typename _CharT, typename _Traits>  
std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> & __os, const  
std::bernoulli_distribution & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>  
std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> & __os, const  
std::geometric_distribution<_IntType> & __x)`
- `template<typename _CharT, typename _Traits>  
std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> & __is,  
std::bernoulli_distribution & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>  
std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> & __is,  
std::geometric_distribution<_IntType> & __x)`

## Detailed Description

### Function Documentation

#### `operator<<()` [1/2]

```

template<typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & std::operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::bernoulli_distribution & __x)

```

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

### Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>bernoulli_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator<<()** [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::geometric_distribution< _IntType > & __x)
```

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>geometric_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

References `std::numeric_limits< _Tp >::epsilon()`, `log()`, and `std::numeric_limits< _Tp >::max()`.

**operator>>()** [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::bernoulli_distribution & __x) [inline]
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                      |
| <code>__x</code>  | A <code>bernoulli_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

References `std::bernoulli_distribution::param()`.

**operator>>()** [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::geometric_distribution< _IntType > & __x)
```

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.



**Parameters**

|                          |                                                          |
|--------------------------|----------------------------------------------------------|
| $\leftrightarrow$<br>_is | An input stream.                                         |
| $\leftrightarrow$<br>_x  | A geometric_distribution random number generator engine. |

**Returns**

The input stream with  $\_x$  extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::geometric\\_distribution<\\_IntType>::param\(\)](#).

**3.11.7.3.3 Normal Distributions**

Collaboration diagram for Normal Distributions:

**Classes**

- class [std::cauchy\\_distribution<\\_RealType>](#)
- class [std::chi\\_squared\\_distribution<\\_RealType>](#)
- class [std::fisher\\_f\\_distribution<\\_RealType>](#)
- class [std::lognormal\\_distribution<\\_RealType>](#)
- class [std::normal\\_distribution<\\_RealType>](#)
- class [std::student\\_t\\_distribution<\\_RealType>](#)

**Functions**

- template<typename \_RealType, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream<\\_CharT, \\_Traits>](#) & [std::operator<<](#) ([std::basic\\_ostream<\\_CharT, \\_Traits>](#) & \_\_os, const [std::cauchy\\_distribution<\\_RealType>](#) & \_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
[std::basic\\_istream<\\_CharT, \\_Traits>](#) & [std::operator>>](#) ([std::basic\\_istream<\\_CharT, \\_Traits>](#) & \_\_is, [std::cauchy\\_distribution<\\_RealType>](#) & \_\_x)

**Detailed Description****Function Documentation****operator<<()**

```

template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & std::operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::cauchy_distribution<_RealType> & __x)

```

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <code>__os</code> | An output stream.                                              |
| <code>__x</code>  | A <code>cauchy_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

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

#### `operator>>()`

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::cauchy_distribution< _RealType > & __x)
Extracts a cauchy_distribution random number distribution __x from the input stream __is.
```

#### Parameters

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                   |
| <code>__x</code>  | A <code>cauchy_distribution</code> random number generator engine. |

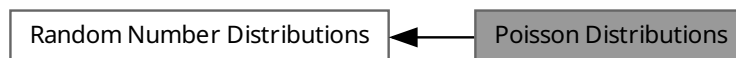
#### Returns

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::cauchy\\_distribution<\\_RealType>::param\(\)](#).

### 3.11.7.3.4 Poisson Distributions

Collaboration diagram for Poisson Distributions:



#### Classes

- class [std::exponential\\_distribution<\\_RealType>](#)
- class [std::extreme\\_value\\_distribution<\\_RealType>](#)
- class [std::gamma\\_distribution<\\_RealType>](#)
- class [std::poisson\\_distribution<\\_IntType>](#)
- class [std::weibull\\_distribution<\\_RealType>](#)

## Functions

- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::weibull_distribution< _RealType > &__x)`

## Detailed Description

### Function Documentation

#### `operator<<()` [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::exponential_distribution< _RealType > & __x)
```

Inserts a `exponential_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                   |
| <code>__x</code>  | A <code>exponential_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

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

#### `operator>>()` [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator>> (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::extreme_value_distribution< _RealType > & __x)
```

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                     |
| <code>__x</code>  | A <code>extreme_value_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

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

**operator<<()** [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::weibull_distribution< _RealType > & __x)
```

Inserts a `weibull_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <code>__os</code> | An output stream.                                               |
| <code>__x</code>  | A <code>weibull_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

References [log\(\)](#), and [pow\(\)](#).

**operator>>()** [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::exponential_distribution< _RealType > & __x)
```

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                         |
|-------------------|-------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                        |
| <code>__x</code>  | A <code>exponential_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::exponential\\_distribution< \\_RealType >::param\(\)](#).

**operator>>()** [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::extreme_value_distribution< _RealType > & __x)
```

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                          |
| <code>__x</code>  | A <code>extreme_value_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

References `std::ios_base::flags()`, and `std::extreme_value_distribution< _RealType >::param()`.

**operator>>()** [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::weibull_distribution< _RealType > & __x)
```

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                    |
| <code>__x</code>  | A <code>weibull_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

References `std::ios_base::flags()`, and `std::weibull_distribution< _RealType >::param()`.

**3.11.7.3.5 Sampling Distributions**

Collaboration diagram for Sampling Distributions:



## Classes

- class [std::discrete\\_distribution<\\_IntType>](#)
- class [std::piecewise\\_constant\\_distribution<\\_RealType>](#)
- class [std::piecewise\\_linear\\_distribution<\\_RealType>](#)

## Detailed Description

### 3.11.7.3.6 Uniform Distributions

Collaboration diagram for Uniform Distributions:



## Concepts

- concept [std::uniform\\_random\\_bit\\_generator](#)

## Classes

- class [std::uniform\\_int\\_distribution<\\_IntType>](#)
- class [std::uniform\\_real\\_distribution<\\_RealType>](#)

## Functions

- [template<typename \\_IntType, typename \\_CharT, typename \\_Traits>  
std::basic\\_ostream<\\_CharT, \\_Traits> & std::operator<< \(std::basic\\_ostream<\\_CharT, \\_Traits> &, const  
std::uniform\\_int\\_distribution<\\_IntType> &\)](#)
- [template<typename \\_RealType, typename \\_CharT, typename \\_Traits>  
std::basic\\_ostream<\\_CharT, \\_Traits> & std::operator<< \(std::basic\\_ostream<\\_CharT, \\_Traits> &, const  
std::uniform\\_real\\_distribution<\\_RealType> &\)](#)
- [template<typename \\_IntType, typename \\_CharT, typename \\_Traits>  
std::basic\\_istream<\\_CharT, \\_Traits> & std::operator>> \(std::basic\\_istream<\\_CharT, \\_Traits> &,  
std::uniform\\_int\\_distribution<\\_IntType> &\)](#)
- [template<typename \\_RealType, typename \\_CharT, typename \\_Traits>  
std::basic\\_istream<\\_CharT, \\_Traits> & std::operator>> \(std::basic\\_istream<\\_CharT, \\_Traits> &,  
std::uniform\\_real\\_distribution<\\_RealType> &\)](#)

## Detailed Description

### Function Documentation

#### `operator<<()` [1/2]

```

template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & std::operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::uniform_int_distribution<_IntType> & __x)

```

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

#### Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                   |
| <code>__x</code>  | A <code>uniform_int_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### `operator<<()` [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::uniform_real_distribution< _RealType > & __x)

```

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                    |
| <code>__x</code>  | A <code>uniform_real_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### `operator>>()` [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::uniform_int_distribution< _IntType > & __x)

```

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

|                   |                                                                         |
|-------------------|-------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                        |
| <code>__x</code>  | A <code>uniform_int_distribution</code> random number generator engine. |

#### Returns

The input stream with `__x` extracted or in an error state.

References `std::ios_base::flags()`, and `std::uniform_int_distribution< _IntType >::param()`.

**operator>>()** [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::uniform_real_distribution< _RealType > & __x)
Extracts a uniform_real_distribution random number distribution __x from the input stream __is.
```

**Parameters**

|                           |                                                             |
|---------------------------|-------------------------------------------------------------|
| $\leftrightarrow$<br>__is | An input stream.                                            |
| $\leftrightarrow$<br>__x  | A uniform_real_distribution random number generator engine. |

**Returns**

The input stream with \_\_x extracted or in an error state.

References [std::ios\\_base::flags\(\)](#), and [std::uniform\\_real\\_distribution< \\_RealType >::param\(\)](#).

**3.11.7.4 Random Number Generators**

Collaboration diagram for Random Number Generators:

**Classes**

- class [std::discard\\_block\\_engine< \\_RandomNumberEngine, \\_\\_p, \\_\\_r >](#)
- class [std::independent\\_bits\\_engine< \\_RandomNumberEngine, \\_\\_w, \\_UIntType >](#)
- class [std::linear\\_congruential\\_engine< \\_UIntType, \\_\\_a, \\_\\_c, \\_\\_m >](#)
- class [std::mersenne\\_twister\\_engine< \\_UIntType, \\_\\_w, \\_\\_n, \\_\\_m, \\_\\_r, \\_\\_a, \\_\\_u, \\_\\_d, \\_\\_s, \\_\\_b, \\_\\_t, \\_\\_c, \\_\\_l, \\_\\_f >](#)
- class [std::philox\\_engine< \\_UIntType, \\_\\_w, \\_\\_n, \\_\\_r, \\_\\_consts >](#)
- class [std::random\\_device](#)
- class [std::shuffle\\_order\\_engine< \\_RandomNumberEngine, \\_\\_k >](#)
- class [std::subtract\\_with\\_carry\\_engine< \\_UIntType, \\_\\_w, \\_\\_s, \\_\\_r >](#)

**Typedefs**

- typedef [minstd\\_rand0](#) [std::default\\_random\\_engine](#)
- typedef [shuffle\\_order\\_engine< minstd\\_rand0, 256 >](#) [std::knuth\\_b](#)
- typedef [linear\\_congruential\\_engine< uint\\_fast32\\_t, 48271UL, 0UL, 2147483647UL >](#) [std::minstd\\_rand](#)
- typedef [linear\\_congruential\\_engine< uint\\_fast32\\_t, 16807UL, 0UL, 2147483647UL >](#) [std::minstd\\_rand0](#)
- typedef [mersenne\\_twister\\_engine< uint\\_fast32\\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >](#) [std::mt19937](#)



- typedef [mersenne\\_twister\\_engine](#)< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [std::mt19937\\_64](#)
- typedef [philox\\_engine](#)< uint\_fast32\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > [std::philox4x32](#)
- typedef [philox\\_engine](#)< uint\_fast64\_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470, EE14C6C93, 0xBB67AE8584CAA73B > [std::philox4x64](#)
- typedef [discard\\_block\\_engine](#)< ranlux24\_base, 223, 23 > [std::ranlux24](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast32\_t, 24, 10, 24 > [std::ranlux24\\_base](#)
- typedef [discard\\_block\\_engine](#)< ranlux48\_base, 389, 11 > [std::ranlux48](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< uint\_fast64\_t, 48, 5, 12 > [std::ranlux48\\_base](#)

## Functions

- template<typename \_RandomNumberEngine, size\_t \_\_w, typename \_UIntType, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [std::operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const  
[std::independent\\_bits\\_engine](#)< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)

### 3.11.7.4.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an operator() that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

**Table 287: Random  
Number  
Gen-  
erator  
Require-  
ments**

|                   |
|-------------------|
| To be documented. |
|-------------------|

### 3.11.7.4.2 Typedef Documentation

#### minstd\_rand

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

#### minstd\_rand0

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

#### mt19937

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL,
7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

### mt19937\_64

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
0x5555555555555555ULL, 17, 0x71d67fffed60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL>
std::mt19937_64
```

An alternative Mersenne Twister.

### philox4x32

```
typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>
std::philox4x32
```

32-bit four-word Philox engine.

### philox4x64

```
typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2↵
E7470EE14C6C93, 0xBB67AE8584CAA73B> std::philox4x64
```

64-bit four-word Philox engine.

## 3.11.7.4.3 Function Documentation

### operator<<()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename
_Traits>
```

```
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)
```

Inserts the current state of a independent\_bits\_engine random number generator engine \_\_x into the output stream \_\_os.

### Parameters

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <code>__os</code> | An output stream.                                         |
| <code>__x</code>  | A independent_bits_engine random number generator engine. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

## 3.11.7.5 Random Number Utilities

Collaboration diagram for Random Number Utilities:



## Classes

- class [std::seed\\_seq](#)

## 3.11.7.5.1 Detailed Description

## 3.11.8 TR1 Mathematical Special Functions

Collaboration diagram for TR1 Mathematical Special Functions:



## Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy>`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)

- float **std::tr1::comp\_ellint\_1f** (float \_\_k)
- long double **std::tr1::comp\_ellint\_1l** (long double \_\_k)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp>::\_\_type **std::tr1::comp\_ellint\_2** (\_Tp \_\_k)
- float **std::tr1::comp\_ellint\_2f** (float \_\_k)
- long double **std::tr1::comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpn>::\_\_type **std::tr1::comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **std::tr1::comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **std::tr1::comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp>::\_\_type **std::tr1::cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp>::\_\_type **std::tr1::cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp>::\_\_type **std::tr1::cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tpnu, \_Tp>::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp>::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Tpp>::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_3<\_Tp, \_Tpn, \_Tpp>::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)

- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 3.11.8.1 Detailed Description

A collection of advanced mathematical special functions.

### 3.11.8.2 Function Documentation

#### assoc\_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (
 unsigned int __n,
 unsigned int __m,
 _Tp __x) [inline]
```

5.2.1.1 Associated Laguerre polynomials.

#### assoc\_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (
 unsigned int __l,
 unsigned int __m,
 _Tp __x) [inline]
```

5.2.1.2 Associated Legendre functions.

#### beta()

```
template<typename _Tpx, typename _Tpy>
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (
 _Tpx __x,
 _Tpy __y) [inline]
```

5.2.1.3 Beta functions.

**comp\_ellint\_1()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (
 _Tp __k) [inline]
```

5.2.1.4 Complete elliptic integrals of the first kind.

**comp\_ellint\_2()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (
 _Tp __k) [inline]
```

5.2.1.5 Complete elliptic integrals of the second kind.

**comp\_ellint\_3()**

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (
 _Tp __k,
 _Tpn __nu) [inline]
```

5.2.1.6 Complete elliptic integrals of the third kind.

**cyl\_bessel\_i()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (
 _Tpnu __nu,
 _Tp __x) [inline]
```

5.2.1.8 Regular modified cylindrical Bessel functions.

**cyl\_bessel\_j()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (
 _Tpnu __nu,
 _Tp __x) [inline]
```

5.2.1.9 Cylindrical Bessel functions (of the first kind).

**cyl\_bessel\_k()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (
 _Tpnu __nu,
 _Tp __x) [inline]
```

5.2.1.10 Irregular modified cylindrical Bessel functions.

**cyl\_neumann()**

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (
 _Tpnu __nu,
 _Tp __x) [inline]
```

5.2.1.11 Cylindrical Neumann functions.

**ellint\_1()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (
 _Tp __k,
 _Tpp __phi) [inline]
```

5.2.1.12 Incomplete elliptic integrals of the first kind.

**ellint\_2()**

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (
 _Tp __k,
 _Tpp __phi) [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

**ellint\_3()**

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (
 _Tp __k,
 _Tpn __nu,
 _Tpp __phi) [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

**expint()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (
 _Tp __x) [inline]
```

5.2.1.15 Exponential integrals.

**hermite()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (
 unsigned int __n,
 _Tp __x) [inline]
```

5.2.1.16 Hermite polynomials.

**laguerre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (
 unsigned int __n,
 _Tp __x) [inline]
```

5.2.1.18 Laguerre polynomials.

**legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (
 unsigned int __n,
 _Tp __x) [inline]
```

5.2.1.19 Legendre polynomials.

**riemann\_zeta()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::riemann_zeta (
 _Tp __x) [inline]
```

5.2.1.20 Riemann zeta function.

**sph\_bessel()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_bessel (
 unsigned int __n,
 _Tp __x) [inline]
```

5.2.1.21 Spherical Bessel functions.

**sph\_legendre()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_legendre (
 unsigned int __l,
 unsigned int __m,
 _Tp __theta) [inline]
```

5.2.1.22 Spherical associated Legendre functions.

**sph\_neumann()**

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::trl::sph_neumann (
 unsigned int __n,
 _Tp __x) [inline]
```

5.2.1.23 Spherical Neumann functions.

## 3.12 Ranges

### Classes

- struct [std::ranges::elements\\_of< \\_Range, \\_Alloc >](#)

#### 3.12.1 Detailed Description

Components for dealing with ranges of elements.



### 3.13 Regular Expressions

Collaboration diagram for Regular Expressions:



#### Topics

- [Base and Implementation Classes](#)

#### Namespaces

- namespace [std::regex\\_constants](#)

#### Classes

- class [std::basic\\_regex](#)< [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::match\\_results](#)< [\\_Bi\\_iter](#), [\\_Alloc](#) >
- class [std::regex\\_error](#)
- class [std::regex\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::regex\\_token\\_iterator](#)< [\\_Bi\\_iter](#), [\\_Ch\\_type](#), [\\_Rx\\_traits](#) >
- class [std::regex\\_traits](#)< [\\_Ch\\_type](#) >
- class [std::sub\\_match](#)< [\\_Bilter](#) >

#### Typedefs

- typedef [match\\_results](#)< const char \* > [std::cmatch](#)
- typedef [regex\\_iterator](#)< const char \* > [std::cregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const char \* > [std::cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [std::csub\\_match](#)
- typedef [basic\\_regex](#)< char > [std::regex](#)
- typedef [match\\_results](#)< string::const\_iterator > [std::smatch](#)
- typedef [regex\\_iterator](#)< string::const\_iterator > [std::sregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< string::const\_iterator > [std::sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< string::const\_iterator > [std::ssub\\_match](#)
- typedef [match\\_results](#)< const wchar\_t \* > [std::wcmatch](#)
- typedef [regex\\_iterator](#)< const wchar\_t \* > [std::wcregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > [std::wcregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const wchar\_t \* > [std::wcs\\_sub\\_match](#)
- typedef [basic\\_regex](#)< wchar\_t > [std::wregex](#)
- typedef [match\\_results](#)< wstring::const\_iterator > [std::wsmatch](#)
- typedef [regex\\_iterator](#)< wstring::const\_iterator > [std::wsregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > [std::wsregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< wstring::const\_iterator > [std::wssub\\_match](#)

## Functions

- `template<typename _ForwardIterator>`  
`std::basic_regex` (`_ForwardIterator`, `_ForwardIterator`, `regex_constants::syntax_option_type`={}) -> `basic_`  
`regex` < `typename iterator_traits` < `_ForwardIterator` > ::`value_type` >
- `template<typename _Bi_iter, typename _Alloc>`  
`bool operator==` (`const match_results` < `_Bi_iter`, `_Alloc` > &\_\_m1, `const match_results` < `_Bi_iter`, `_Alloc` > &\_\_m2)
- `template<typename _Ch_type, typename _Rx_traits>`  
`void swap` (`basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_lhs, `basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_rhs) noexcept
- `template<typename _Bi_iter, typename _Alloc>`  
`void swap` (`match_results` < `_Bi_iter`, `_Alloc` > &\_\_lhs, `match_results` < `_Bi_iter`, `_Alloc` > &\_\_rhs) noexcept

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`_Bi_iter` \_\_s, `_Bi_iter` \_\_e, `match_results` < `_Bi_iter`, `_Alloc` > &\_\_m, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`_Bi_iter` \_\_first, `_Bi_iter` \_\_last, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits>`  
`bool std::regex_match` (`const _Ch_type` \*\_\_s, `match_results` < `const _Ch_type` \*, `_Alloc` > &\_\_m, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_f=`regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string` < `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > &\_\_s, `match_results` < `typename basic_string` < `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > ::`const_iterator`, `_Alloc` > &\_\_m, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string` < `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > &&, `match_results` < `typename basic_string` < `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > ::`const_iterator`, `_Alloc` > &, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &, `regex_constants::match_flag_type`=`regex_constants::match_default`)=delete
- `template<typename _Ch_type, class _Rx_traits>`  
`bool std::regex_match` (`const _Ch_type` \*\_\_s, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_f=`regex_constants::match_default`)
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match` (`const basic_string` < `_Ch_type`, `_Ch_traits`, `_Str_allocator` > &\_\_s, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`_Bi_iter` \_\_s, `_Bi_iter` \_\_e, `match_results` < `_Bi_iter`, `_Alloc` > &\_\_m, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`_Bi_iter` \_\_first, `_Bi_iter` \_\_last, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_re, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)
- `template<typename _Ch_type, class _Alloc, class _Rx_traits>`  
`bool std::regex_search` (`const _Ch_type` \*\_\_s, `match_results` < `const _Ch_type` \*, `_Alloc` > &\_\_m, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_e, `regex_constants::match_flag_type` \_\_f=`regex_constants::match_default`)
- `template<typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`const _Ch_type` \*\_\_s, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_e, `regex_constants::match_flag_type` \_\_f=`regex_constants::match_default`)
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search` (`const basic_string` < `_Ch_type`, `_Ch_traits`, `_String_allocator` > &\_\_s, `const basic_regex` < `_Ch_type`, `_Rx_traits` > &\_\_e, `regex_constants::match_flag_type` \_\_flags=`regex_constants::match_default`)

- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↵`  
`Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↵`  
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx↵`  
`_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits`  
`> &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bilter>`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter>`  
`auto operator<=> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs) noexcept(__↵`  
`detail::__is_contiguous_iter< _Bilter >::value)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↵`  
`_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↵`  
`_Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`* __rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`* __rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`& __rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const`  
`& __rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`

- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>`  
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > & __os, const`  
`sub_match< _Bi_iter > & __m)`

### 3.13.1 Detailed Description

A facility for performing regular expression pattern matching.

Since

C++11

### 3.13.2 Typedef Documentation

#### **cregex\_token\_iterator**

typedef `regex_token_iterator<const char*>` `std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

#### **csub\_match**

typedef `sub_match<const char*>` `std::csub_match`

Standard regex submatch over a C-style null-terminated string.

#### **regex**

typedef `basic_regex<char>` `std::regex`

Standard regular expressions.

#### **sregex\_token\_iterator**

typedef `regex_token_iterator<string::const_iterator>` `std::sregex_token_iterator`

Token iterator for standard strings.

#### **ssub\_match**

typedef `sub_match<string::const_iterator>` `std::ssub_match`

Standard regex submatch over a standard string.

#### **wcregex\_token\_iterator**

typedef `regex_token_iterator<const wchar_t*>` `std::wcregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

#### **wcsub\_match**

typedef `sub_match<const wchar_t*>` `std::wcsub_match`

Regex submatch over a C-style null-terminated wide string.

#### **wregex**

typedef `basic_regex<wchar_t>` `std::wregex`

Standard wide-character regular expressions.

**wsregex\_token\_iterator**

typedef `regex_token_iterator`<wstring::const\_iterator> `std::wsregex_token_iterator`  
 Token iterator for standard wide-character strings.

**wssub\_match**

typedef `sub_match`<wstring::const\_iterator> `std::wssub_match`  
 Regex submatch over a standard wide string.

**3.13.3 Function Documentation****operator<<()**

```
template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
basic_ostream< _Ch_type, _Ch_traits > & operator<< (
 basic_ostream< _Ch_type, _Ch_traits > & __os,
 const sub_match< _Bi_iter > & __m) [related]
```

Inserts a matched string into an output stream.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__os</code> | The output stream. |
| <code>__m</code>  | A submatch string. |

**Returns**

the output stream with the submatch string inserted.

**operator<=>() [1/4]**

```
template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
auto operator<=> (
 const sub_match< _Bi_iter > & __lhs,
 const __sub_match_string< _Bi_iter, _Ch_traits, _Alloc > & __rhs) [related]
```

Three-way comparison of a regular expression submatch and a string.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A string.                      |

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [2/4]**

```
template<typename _Bi_iter>
auto operator<=> (
```

```
const sub_match< _Bi_iter > & __lhs,
typename iterator_traits< _Bi_iter >::value_type const & __rhs) [related]
```

Three-way comparison of a regular expression submatch and a character.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A character.                   |

#### Returns

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

#### `operator<=>()` [3/4]

```
template<typename _Bi_iter>
auto operator<=> (
 const sub_match< _Bi_iter > & __lhs,
 typename iterator_traits< _Bi_iter >::value_type const * __rhs) [related]
```

Three-way comparison of a regular expression submatch and a C string.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A null-terminated string.      |

#### Returns

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

#### `operator<=>()` [4/4]

```
template<typename _BiIter>
auto operator<=> (
 const sub_match< _BiIter > & __lhs,
 const sub_match< _BiIter > & __rhs) [related]
```

Three-way comparison of two regular expression submatches.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__lhs</code> | First regular expression submatch.  |
| <code>__rhs</code> | Second regular expression submatch. |

#### Returns

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator==()** [1/5]

```
template<typename _Bi_iter, typename _Alloc>
bool operator== (
 const match_results< _Bi_iter, _Alloc > & __m1,
 const match_results< _Bi_iter, _Alloc > & __m2) [related]
```

Compares two `match_results` for equality.

**Returns**

true if the two objects refer to the same match, false otherwise.

**operator==()** [2/5]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
bool operator== (
 const sub_match< _Bi_iter > & __lhs,
 const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a string.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A string.                      |

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**operator==()** [3/5]

```
template<typename _Bi_iter>
bool operator== (
 const sub_match< _Bi_iter > & __lhs,
 typename iterator_traits< _Bi_iter >::value_type const & __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a character.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A character.                   |

**Returns**

true if `__lhs` is equivalent to `__rhs`, false otherwise.

**operator==()** [4/5]

```
template<typename _Bi_iter>
bool operator== (
 const sub_match< _Bi_iter > & __lhs,
 typename iterator_traits< _Bi_iter >::value_type const * __rhs) [related]
```

Tests the equivalence of a regular expression submatch and a C string.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__lhs</code> | A regular expression submatch. |
| <code>__rhs</code> | A null-terminated string.      |

#### Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

#### `operator==( )` [5/5]

```
template<typename _BiIter>
bool operator== (
 const sub_match< _BiIter > & __lhs,
 const sub_match< _BiIter > & __rhs) [related]
```

Tests the equivalence of two regular expression submatches.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__lhs</code> | First regular expression submatch.  |
| <code>__rhs</code> | Second regular expression submatch. |

#### Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

#### `regex_match()` [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
 _Bi_iter __first,
 _Bi_iter __last,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

#### Parameters

|                      |                                                      |
|----------------------|------------------------------------------------------|
| <code>__first</code> | Beginning of the character sequence to match.        |
| <code>__last</code>  | One-past-the-end of the character sequence to match. |
| <code>__re</code>    | The regular expression.                              |
| <code>__flags</code> | Controls how the regular expression is matched.      |



**Return values**

|              |                 |
|--------------|-----------------|
| <i>true</i>  | A match exists. |
| <i>false</i> | Otherwise.      |

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
 _Bi_iter __s,
 _Bi_iter __e,
 match_results< _Bi_iter, _Alloc > & __m,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

**Parameters**

|                      |                                                      |
|----------------------|------------------------------------------------------|
| <code>__s</code>     | Start of the character sequence to match.            |
| <code>__e</code>     | One-past-the-end of the character sequence to match. |
| <code>__m</code>     | The match results.                                   |
| <code>__re</code>    | The regular expression.                              |
| <code>__flags</code> | Controls how the regular expression is matched.      |

**Return values**

|              |                 |
|--------------|-----------------|
| <i>true</i>  | A match exists. |
| <i>false</i> | Otherwise.      |

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#).

Referenced by [regex\\_match\(\)](#), [regex\\_match\(\)](#), [regex\\_match\(\)](#), [regex\\_match\(\)](#), and [regex\\_match\(\)](#).

**regex\_match()** [3/7]

```
template<typename _Ch_type, class _Rx_traits>
bool std::regex_match (
 const _Ch_type * __s,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression *e* and a C-style null-terminated string.

**Parameters**

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| $\leftrightarrow$<br><i>__s</i>  | The C-style null-terminated string to match.    |
| $\leftrightarrow$<br><i>__re</i> | The regular expression.                         |
| $\leftrightarrow$<br><i>__f</i>  | Controls how the regular expression is matched. |

**Return values**

|              |                 |
|--------------|-----------------|
| <i>true</i>  | A match exists. |
| <i>false</i> | Otherwise.      |

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

**regex\_match()** [4/7]

```
template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
bool std::regex_match (
 const _Ch_type * __s,
 match_results< const _Ch_type *, _Alloc > & __m,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression *e* and a C-style null-terminated string.

**Parameters**

|                                 |                                              |
|---------------------------------|----------------------------------------------|
| $\leftrightarrow$<br><i>__s</i> | The C-style null-terminated string to match. |
| $\leftrightarrow$<br><i>__m</i> | The match results.                           |

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>_↵<br/>_re</code> | The regular expression.                         |
| <code>_↵<br/>_f</code>  | Controls how the regular expression is matched. |

### Return values

|                    |                 |
|--------------------|-----------------|
| <code>true</code>  | A match exists. |
| <code>false</code> | Otherwise.      |

### Exceptions

|                 |                                              |
|-----------------|----------------------------------------------|
| <code>an</code> | exception of type <code>regex_error</code> . |
|-----------------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

### regex\_match() [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↵
Rx_traits>
bool std::regex_match (
 const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
 match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & ,
 const basic_regex< _Ch_type, _Rx_traits > & ,
 regex_constants::match_flag_type = regex_constants::match_default) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

References [std::regex\\_constants::match\\_default](#).

### regex\_match() [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↵
Rx_traits>
bool std::regex_match (
 const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
 match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & __m,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Determines if there is a match between the regular expression `e` and a string.

### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The string to match. |
|------------------|----------------------|

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>__m</code>     | The match results.                              |
| <code>__re</code>    | The regular expression.                         |
| <code>__flags</code> | Controls how the regular expression is matched. |

### Return values

|                    |                 |
|--------------------|-----------------|
| <code>true</code>  | A match exists. |
| <code>false</code> | Otherwise.      |

### Exceptions

|                 |                                              |
|-----------------|----------------------------------------------|
| <code>an</code> | exception of type <code>regex_error</code> . |
|-----------------|----------------------------------------------|

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

### regex\_match() [7/7]

```
template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
 const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
 const basic_regex< _Ch_type, _Rx_traits > & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Indicates if there is a match between the regular expression `e` and a string.

### Parameters

|                      |                                                      |
|----------------------|------------------------------------------------------|
| <code>__s</code>     | [IN] The string to match.                            |
| <code>__re</code>    | [IN] The regular expression.                         |
| <code>__flags</code> | [IN] Controls how the regular expression is matched. |

### Return values

|                    |                 |
|--------------------|-----------------|
| <code>true</code>  | A match exists. |
| <code>false</code> | Otherwise.      |

### Exceptions

|                 |                                              |
|-----------------|----------------------------------------------|
| <code>an</code> | exception of type <code>regex_error</code> . |
|-----------------|----------------------------------------------|

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_match\(\)](#).

### regex\_replace() [1/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
_Out_iter std::regex_replace (
 _Out_iter __out,
 _Bi_iter __first,
 _Bi_iter __last,
 const basic_regex<_Ch_type, _Rx_traits> & __e,
 const _Ch_type * __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default)
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

#### Parameters

|                      |                                                |
|----------------------|------------------------------------------------|
| <code>__out</code>   | [OUT] The output iterator.                     |
| <code>__first</code> | [IN] The start of the string to search.        |
| <code>__last</code>  | [IN] One-past-the-end of the string to search. |
| <code>__e</code>     | [IN] The regular expression to search for.     |
| <code>__fmt</code>   | [IN] The format C-string.                      |
| <code>__flags</code> | [IN] Search and replace policy flags.          |

#### Returns

`__out`

#### Exceptions

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#).

### regex\_replace() [2/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename
_St, typename _Sa>
_Out_iter std::regex_replace (
 _Out_iter __out,
 _Bi_iter __first,
 _Bi_iter __last,
 const basic_regex<_Ch_type, _Rx_traits> & __e,
 const basic_string<_Ch_type, _St, _Sa> & __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

**Parameters**

|                      |                                                |
|----------------------|------------------------------------------------|
| <code>__out</code>   | [OUT] The output iterator.                     |
| <code>__first</code> | [IN] The start of the string to search.        |
| <code>__last</code>  | [IN] One-past-the-end of the string to search. |
| <code>__e</code>     | [IN] The regular expression to search for.     |
| <code>__fmt</code>   | [IN] The format string.                        |
| <code>__flags</code> | [IN] Search and replace policy flags.          |

**Returns**`__out`**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::length\(\)](#), and [std::regex\\_constants::match\\_default](#).

Referenced by [regex\\_replace\(\)](#), [regex\\_replace\(\)](#), [regex\\_replace\(\)](#), and [regex\\_replace\(\)](#).

**regex\_replace() [3/6]**

```
template<typename _Rx_traits, typename _Ch_type>
basic_string<_Ch_type> std::regex_replace (
 const _Ch_type * __s,
 const basic_regex<_Ch_type, _Rx_traits> & __e,
 const _Ch_type * __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__s</code>     | [IN] The C-string to search and replace.   |
| <code>__e</code>     | [IN] The regular expression to search for. |
| <code>__fmt</code>   | [IN] The format C-string.                  |
| <code>__flags</code> | [IN] Search and replace policy flags.      |

**Returns**

The string after replacing.

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [back\\_inserter\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [4/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type > std::regex_replace (
 const _Ch_type * __s,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 const basic_string< _Ch_type, _St, _Sa > & __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__s</code>     | [IN] The C-string to search and replace.   |
| <code>__e</code>     | [IN] The regular expression to search for. |
| <code>__fmt</code>   | [IN] The format string.                    |
| <code>__flags</code> | [IN] Search and replace policy flags.      |

**Returns**

The string after replacing.

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [back\\_inserter\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [5/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
 const basic_string< _Ch_type, _St, _Sa > & __s,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 const _Ch_type * __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__s</code>     | [IN] The string to search and replace.     |
| <code>__e</code>     | [IN] The regular expression to search for. |
| <code>__fmt</code>   | [IN] The format C-string.                  |
| <code>__flags</code> | [IN] Search and replace policy flags.      |

**Returns**

The string after replacing.

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [back\\_inserter\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

**regex\_replace()** [6/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename
_Fsa>
basic_string<_Ch_type, _St, _Sa> std::regex_replace (
 const basic_string<_Ch_type, _St, _Sa> & __s,
 const basic_regex<_Ch_type, _Rx_traits> & __e,
 const basic_string<_Ch_type, _Fst, _Fsa> & __fmt,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__s</code>     | [IN] The string to search and replace.     |
| <code>__e</code>     | [IN] The regular expression to search for. |
| <code>__fmt</code>   | [IN] The format string.                    |
| <code>__flags</code> | [IN] Search and replace policy flags.      |

**Returns**

The string after replacing.

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|



References [back\\_inserter\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::begin\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_replace\(\)](#).

### regex\_search() [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
 _Bi_iter __first,
 _Bi_iter __last,
 const basic_regex<_Ch_type, _Rx_traits> & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a range.

#### Parameters

|                |                                                |
|----------------|------------------------------------------------|
| <i>__first</i> | [IN] The start of the string to search.        |
| <i>__last</i>  | [IN] One-past-the-end of the string to search. |
| <i>__re</i>    | [IN] The regular expression to search for.     |
| <i>__flags</i> | [IN] Search policy flags.                      |

#### Return values

|              |                                       |
|--------------|---------------------------------------|
| <i>true</i>  | A match was found within the string.  |
| <i>false</i> | No match was found within the string. |

#### Exceptions

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

### regex\_search() [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
 _Bi_iter __s,
 _Bi_iter __e,
 match_results<_Bi_iter, _Alloc> & __m,
 const basic_regex<_Ch_type, _Rx_traits> & __re,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a range.

#### Parameters

|            |                                         |
|------------|-----------------------------------------|
| <i>__s</i> | [IN] The start of the string to search. |
|------------|-----------------------------------------|

|                      |                                                |
|----------------------|------------------------------------------------|
| <code>__e</code>     | [IN] One-past-the-end of the string to search. |
| <code>__m</code>     | [OUT] The match results.                       |
| <code>__re</code>    | [IN] The regular expression to search for.     |
| <code>__flags</code> | [IN] Search policy flags.                      |

### Return values

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| <code>true</code>  | A match was found within the string.                                              |
| <code>false</code> | No match was found within the string, the content of <code>m</code> is undefined. |

### Exceptions

|                 |                                              |
|-----------------|----------------------------------------------|
| <code>an</code> | exception of type <code>regex_error</code> . |
|-----------------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#).

Referenced by [std::regex\\_iterator< const char \\* >::regex\\_iterator\(\)](#), [std::regex\\_iterator< \\_Bi\\_iter, \\_Ch\\_type, \\_Rx\\_traits >::operator++\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), [regex\\_search\(\)](#), and [regex\\_search\(\)](#).

### regex\_search() [3/7]

```
template<typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
 const _Ch_type * __s,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__s</code> | [IN] The C-string to search.               |
| <code>__e</code> | [IN] The regular expression to search for. |
| <code>__f</code> | [IN] Search policy flags.                  |

### Return values

|                   |                                      |
|-------------------|--------------------------------------|
| <code>true</code> | A match was found within the string. |
|-------------------|--------------------------------------|

|              |                                       |
|--------------|---------------------------------------|
| <i>false</i> | No match was found within the string. |
|--------------|---------------------------------------|

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [4/7]

```
template<typename _Ch_type, class _Alloc, class _Rx_traits>
bool std::regex_search (
 const _Ch_type * __s,
 match_results< const _Ch_type *, _Alloc > & __m,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 regex_constants::match_flag_type __f = regex_constants::match_default) [inline]
```

Searches for a regular expression within a C-string.

**Parameters**

|                           |                                            |
|---------------------------|--------------------------------------------|
| $\leftarrow$<br><i>_s</i> | [IN] A C-string to search for the regex.   |
| $\leftarrow$<br><i>_m</i> | [OUT] The set of regex matches.            |
| $\leftarrow$<br><i>_e</i> | [IN] The regex to search for in <i>s</i> . |
| $\leftarrow$<br><i>_f</i> | [IN] The search flags.                     |

**Return values**

|              |                                                                             |
|--------------|-----------------------------------------------------------------------------|
| <i>true</i>  | A match was found within the string.                                        |
| <i>false</i> | No match was found within the string, the content of <i>m</i> is undefined. |

**Exceptions**

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename \leftarrow
_Rx_traits>
```

```

bool std::regex_search (
 const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
 match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & ,
 const basic_regex< _Ch_type, _Rx_traits > & ,
 regex_constants::match_flag_type = regex_constants::match_default) [delete]

```

Prevent unsafe attempts to get match\_results from a temporary string.

References [std::regex\\_constants::match\\_default](#).

### regex\_search() [6/7]

```

template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
 const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
 match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 regex_constants::match_flag_type __f = regex_constants::match_default) [inline]

```

Searches for a regular expression within a string.

#### Parameters

|                                  |                                            |
|----------------------------------|--------------------------------------------|
| <a href="#"><code>__s</code></a> | [IN] A C++ string to search for the regex. |
| <a href="#"><code>__m</code></a> | [OUT] The set of regex matches.            |
| <a href="#"><code>__e</code></a> | [IN] The regex to search for in s.         |
| <a href="#"><code>__f</code></a> | [IN] The search flags.                     |

#### Return values

|              |                                                                      |
|--------------|----------------------------------------------------------------------|
| <i>true</i>  | A match was found within the string.                                 |
| <i>false</i> | No match was found within the string, the content of m is undefined. |

#### Exceptions

|           |                                              |
|-----------|----------------------------------------------|
| <i>an</i> | exception of type <code>regex_error</code> . |
|-----------|----------------------------------------------|

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::begin\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::end\(\)](#), [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

**regex\_search()** [7/7]

```
template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
 const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
 const basic_regex< _Ch_type, _Rx_traits > & __e,
 regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]
```

Searches for a regular expression within a string.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__s</code>     | [IN] The string to search.                 |
| <code>__e</code>     | [IN] The regular expression to search for. |
| <code>__flags</code> | [IN] Search policy flags.                  |

**Return values**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>true</code>  | A match was found within the string.  |
| <code>false</code> | No match was found within the string. |

**Exceptions**

|                 |                                              |
|-----------------|----------------------------------------------|
| <code>an</code> | exception of type <code>regex_error</code> . |
|-----------------|----------------------------------------------|

References [std::regex\\_constants::match\\_default](#), and [regex\\_search\(\)](#).

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

```
template<typename _Ch_type, typename _Rx_traits>
void swap (
 basic_regex< _Ch_type, _Rx_traits > & __lhs,
 basic_regex< _Ch_type, _Rx_traits > & __rhs) [related]
```

Swaps the contents of two regular expression objects.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__lhs</code> | First regular expression.  |
| <code>__rhs</code> | Second regular expression. |

**swap()** [2/2]

```
template<typename _Bi_iter, typename _Alloc>
void swap (
 match_results< _Bi_iter, _Alloc > & __lhs,
 match_results< _Bi_iter, _Alloc > & __rhs) [related]
```

Swaps two match results.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__lhs</code> | A match result. |
| <code>__rhs</code> | A match result. |

The contents of the two `match_results` objects are swapped.

**3.13.4 Base and Implementation Classes**

Collaboration diagram for Base and Implementation Classes:

**Classes**

- struct `std::__detail::BracketMatcher<_TraitsT, __icase, __collate >`
- class `std::__detail::Compiler<_TraitsT >`
- class `std::__detail::Executor<_Bilter, _Alloc, _TraitsT >`
- class `std::__detail::Scanner<_CharT >`
- class `std::__detail::StateSeq<_TraitsT >`

**Typedefs**

- template<typename `_CharT`>  
using `std::__detail::Matcher`
- typedef long `std::__detail::StateldT`

**Enumerations**

- enum `std::__detail::Opcode` : int {  
`_S_opcode_unknown` , `_S_opcode_alternative` , `_S_opcode_repeat` , `_S_opcode_backref` ,  
`_S_opcode_line_begin_assertion` , `_S_opcode_line_end_assertion` , `_S_opcode_word_boundary` , `_S_opcode_subexpr_lookahead` ,  
`_S_opcode_subexpr_begin` , `_S_opcode_subexpr_end` , `_S_opcode_dummy` , `_S_opcode_match` ,  
`_S_opcode_accept` }

**Variables**

- constexpr `_StateldT std::__detail::_S_invalid_state_id`

### 3.13.4.1 Detailed Description

### 3.13.4.2 Enumeration Type Documentation

#### `_Opcode`

```
enum std::__detail::_Opcode : int
```

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

## 3.14 Strings

Collaboration diagram for Strings:



### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::basic_string_view<_CharT, _Traits>`
- class `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>`
- struct `std::char_traits<_CharT>`

### Typedefs

- typedef `basic_string<char>` `std::string`
- typedef `basic_string<char16_t>` `std::u16string`
- typedef `basic_string<char32_t>` `std::u32string`
- typedef `basic_string<wchar_t>` `std::wstring`

### 3.14.1 Detailed Description

### 3.14.2 Typedef Documentation

#### `string`

```
typedef basic_string<char> std::string
```

A string of `char`.

### u16string

```
typedef basic_string<char16_t> std::u16string
```

A string of `char16_t`.

### u32string

```
typedef basic_string<char32_t> std::u32string
```

A string of `char32_t`.

### wstring

```
typedef basic_string<wchar_t> std::wstring
```

A string of `wchar_t`.

## 3.15 Technical Specifications

Collaboration diagram for Technical Specifications:



### Topics

- [Filesystem TS](#)
- [Library Fundamentals TS](#)
- [Parallelism TS](#)

#### 3.15.1 Detailed Description

Components specified by various Technical Specifications.

As indicated by the `std::experimental` namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.



### 3.15.2 Filesystem TS

Collaboration diagram for Filesystem TS:



#### Files

- file [filesystem](#)

#### Classes

- class [std::experimental::filesystem::v1::filesystem\\_error](#)
- class [std::experimental::filesystem::v1::path::iterator](#)
- class [std::experimental::filesystem::v1::path](#)
- struct [std::experimental::filesystem::v1::space\\_info](#)

#### Typedefs

- using [std::experimental::filesystem::file\\_time\\_type](#)

#### Enumerations

- enum class [std::experimental::filesystem::copy\\_options](#) : unsigned short { **none** , **skip\_existing** , **overwrite\_existing** , **update\_existing** , **recursive** , **copy\_symlinks** , **skip\_symlinks** , **directories\_only** , **create\_symlinks** , **create\_hard\_links** }
- enum class [std::experimental::filesystem::directory\\_options](#) : unsigned char { **none** , **follow\_directory\_symlink** , **skip\_permission\_denied** }
- enum class [std::experimental::filesystem::file\\_type](#) : signed char { **none** , **not\_found** , **regular** , **directory** , **symlink** , **block** , **character** , **fifo** , **socket** , **unknown** }
- enum class [std::experimental::filesystem::perms](#) : unsigned { **none** , **owner\_read** , **owner\_write** , **owner\_exec** , **owner\_all** , **group\_read** , **group\_write** , **group\_exec** , **group\_all** , **others\_read** , **others\_write** , **others\_exec** , **others\_all** , **all** , **set\_uid** , **set\_gid** , **sticky\_bit** , **mask** , **unknown** , **add\_perms** , **remove\_perms** , **symlink\_nofollow** }

#### Functions

- [std::experimental::filesystem::v1::path::path](#) (const path &\_\_p)
- [std::experimental::filesystem::v1::path::path](#) (path &&\_\_p) noexcept
- [std::experimental::filesystem::v1::path::path](#) (string\_type &&\_\_source)

- `path std::experimental::filesystem::absolute` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path & std::experimental::filesystem::v1::path::assign` (string\_type &&\_\_source)
- `directory_iterator std::experimental::filesystem::begin` (directory\_iterator \_\_iter) noexcept
- `recursive_directory_iterator std::experimental::filesystem::begin` (recursive\_directory\_iterator \_\_iter) noexcept
- `iterator std::experimental::filesystem::v1::path::begin` () const noexcept
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- `int std::experimental::filesystem::v1::path::compare` (const `basic_string_view`< value\_type > \_\_s) const
- `int std::experimental::filesystem::v1::path::compare` (const string\_type &\_\_s) const
- `int std::experimental::filesystem::v1::path::compare` (const value\_type \* \_\_s) const
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_options)
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_options, `error_code` &) noexcept
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to)
- `bool std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_option)
- `bool std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_option, `error_code` &)
- `bool std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec)
- `void std::experimental::filesystem::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_↵ symlink)
- `void std::experimental::filesystem::copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_↵ symlink, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::create_directories` (const `path` &\_\_p)
- `bool std::experimental::filesystem::create_directories` (const `path` &\_\_p, `error_code` &\_\_ec)
- `bool std::experimental::filesystem::create_directory` (const `path` &\_\_p)
- `bool std::experimental::filesystem::create_directory` (const `path` &\_\_p, const `path` &\_\_attributes)
- `bool std::experimental::filesystem::create_directory` (const `path` &\_\_p, const `path` &\_\_attributes, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::create_directory` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- `void std::experimental::filesystem::create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link)
- `void std::experimental::filesystem::create_hard_link` (const `path` &\_\_to, const `path` &\_\_new\_hard\_link, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- `void std::experimental::filesystem::create_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- `path std::experimental::filesystem::current_path` ()
- `void std::experimental::filesystem::current_path` (const `path` &\_\_p)
- `void std::experimental::filesystem::current_path` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `path std::experimental::filesystem::current_path` (`error_code` &\_\_ec)
- `directory_iterator std::experimental::filesystem::end` (directory\_iterator) noexcept
- `recursive_directory_iterator std::experimental::filesystem::end` (recursive\_directory\_iterator) noexcept
- `iterator std::experimental::filesystem::v1::path::end` () const noexcept
- `bool std::experimental::filesystem::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2)
- `bool std::experimental::filesystem::equivalent` (const `path` &\_\_p1, const `path` &\_\_p2, `error_code` &\_\_ec) noexcept

- `bool std::experimental::filesystem::exists (const path &__p)`
- `bool std::experimental::filesystem::exists (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::exists (file_status __s) noexcept`
- `path std::experimental::filesystem::v1::path::extension () const`
- `uintmax_t std::experimental::filesystem::file_size (const path &__p)`
- `uintmax_t std::experimental::filesystem::file_size (const path &__p, error_code &__ec) noexcept`
- `path std::experimental::filesystem::v1::path::filename () const`
- `std::string std::experimental::filesystem::v1::path::generic_string () const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>  
std::basic_string< _CharT, _Traits, _Allocator > std::experimental::filesystem::v1::path::generic_string  
(const _Allocator &__a=_Allocator()) const`
- `std::u16string std::experimental::filesystem::v1::path::generic_u16string () const`
- `std::u32string std::experimental::filesystem::v1::path::generic_u32string () const`
- `std::string std::experimental::filesystem::v1::path::generic_u8string () const`
- `std::wstring std::experimental::filesystem::v1::path::generic_wstring () const`
- `uintmax_t std::experimental::filesystem::hard_link_count (const path &__p)`
- `uintmax_t std::experimental::filesystem::hard_link_count (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::v1::path::has_extension () const`
- `bool std::experimental::filesystem::v1::path::has_stem () const`
- `bool std::experimental::filesystem::v1::path::is_absolute () const`
- `bool std::experimental::filesystem::is_block_file (const path &__p)`
- `bool std::experimental::filesystem::is_block_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_block_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_character_file (const path &__p)`
- `bool std::experimental::filesystem::is_character_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_character_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_directory (const path &__p)`
- `bool std::experimental::filesystem::is_directory (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_directory (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_empty (const path &__p)`
- `bool std::experimental::filesystem::is_empty (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_fifo (const path &__p)`
- `bool std::experimental::filesystem::is_fifo (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_fifo (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_other (const path &__p)`
- `bool std::experimental::filesystem::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_other (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_regular_file (const path &__p)`
- `bool std::experimental::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_regular_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_socket (const path &__p)`
- `bool std::experimental::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_socket (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_symlink (const path &__p)`
- `bool std::experimental::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_symlink (file_status __s) noexcept`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p)`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`

- `path & std::experimental::filesystem::v1::path::make_preferred ()`
- `bool std::experimental::filesystem::operator!= (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator!= (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `copy_options & std::experimental::filesystem::operator&= (copy_options &__x, copy_options __y) noexcept`
- `reference std::experimental::filesystem::v1::path::iterator::operator* () const noexcept`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator++ () noexcept`
- `template<typename _CharT>  
__detail::_Path< _CharT *, _CharT * > & std::experimental::filesystem::v1::path::operator+= (_CharT __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (basic_string_view< value_type > __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const path &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const string_type &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const value_type * __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (value_type __x)`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator-- () noexcept`
- `bool std::experimental::filesystem::operator< (const path &__lhs, const path &__rhs) noexcept`
- `path & std::experimental::filesystem::v1::path::operator= (const path &__p)`
- `path & std::experimental::filesystem::v1::path::operator= (path &&__p) noexcept`
- `path & std::experimental::filesystem::v1::path::operator= (string_type &&__source)`
- `bool std::experimental::filesystem::operator== (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator== (const path &__lhs, const path &__rhs) noexcept`
- `bool std::experimental::filesystem::operator== (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `constexpr copy_options std::experimental::filesystem::operator^ (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::experimental::filesystem::operator^= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::operator| (copy_options __x, copy_options __y) noexcept`
- `copy_options & std::experimental::filesystem::operator|= (copy_options &__x, copy_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::operator~ (copy_options __x) noexcept`
- `void std::experimental::filesystem::permissions (const path &__p, perms __prms)`
- `void std::experimental::filesystem::permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `path std::experimental::filesystem::read_symlink (const path &__p)`
- `path std::experimental::filesystem::read_symlink (const path &__p, error_code &__ec)`
- `bool std::experimental::filesystem::remove (const path &__p)`
- `bool std::experimental::filesystem::remove (const path &__p, error_code &__ec) noexcept`
- `uintmax_t std::experimental::filesystem::remove_all (const path &__p)`
- `uintmax_t std::experimental::filesystem::remove_all (const path &__p, error_code &__ec)`
- `void std::experimental::filesystem::rename (const path &__from, const path &__to)`
- `void std::experimental::filesystem::rename (const path &__from, const path &__to, error_code &__ec) noexcept`
- `void std::experimental::filesystem::resize_file (const path &__p, uintmax_t __size)`
- `void std::experimental::filesystem::resize_file (const path &__p, uintmax_t __size, error_code &__ec) noexcept`
- `space_info std::experimental::filesystem::space (const path &__p)`
- `space_info std::experimental::filesystem::space (const path &__p, error_code &__ec) noexcept`
- `file_status std::experimental::filesystem::status (const path &__p)`
- `file_status std::experimental::filesystem::status (const path &__p, error_code &__ec) noexcept`

- `bool std::experimental::filesystem::status_known (file_status) noexcept`
- `path std::experimental::filesystem::v1::path::stem () const`
- `std::string std::experimental::filesystem::v1::path::string () const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>  
std::basic_string<_CharT, _Traits, _Allocator> std::experimental::filesystem::v1::path::string (const _↵  
Allocator &__a=_Allocator()) const`
- `void std::experimental::filesystem::v1::path::swap (path &__rhs) noexcept`
- `file_status std::experimental::filesystem::symlink_status (const path &)`
- `file_status std::experimental::filesystem::symlink_status (const path &, error_code &) noexcept`
- `path std::experimental::filesystem::system_complete (const path &__p)`
- `path std::experimental::filesystem::system_complete (const path &__p, error_code &__ec)`
- `path std::experimental::filesystem::temp_directory_path ()`
- `path std::experimental::filesystem::temp_directory_path (error_code &__ec)`
- `std::u16string std::experimental::filesystem::v1::path::u16string () const`
- `std::u32string std::experimental::filesystem::v1::path::u32string () const`
- `std::string std::experimental::filesystem::v1::path::u8string () const`
- `std::wstring std::experimental::filesystem::v1::path::wstring () const`
- `constexpr perms std::experimental::filesystem::operator| (perms __x, perms __y) noexcept`
- `constexpr perms std::experimental::filesystem::operator^ (perms __x, perms __y) noexcept`
- `constexpr perms std::experimental::filesystem::operator~ (perms __x) noexcept`
- `perms & std::experimental::filesystem::operator&= (perms &__x, perms __y) noexcept`
- `perms & std::experimental::filesystem::operator|= (perms &__x, perms __y) noexcept`
- `perms & std::experimental::filesystem::operator^= (perms &__x, perms __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator| (directory_options __x, directory_options  
__y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator^ (directory_options __x, directory_↵  
options __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator~ (directory_options __x) noexcept`
- `directory_options & std::experimental::filesystem::operator&= (directory_options &__x, directory_options _↵  
__y) noexcept`
- `directory_options & std::experimental::filesystem::operator|= (directory_options &__x, directory_options __y)  
noexcept`
- `directory_options & std::experimental::filesystem::operator^= (directory_options &__x, directory_options _↵  
__y) noexcept`

### 3.15.2.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories.  
ISO/IEC TS 18822:2015 C++ File System Technical Specification

Since

C++11

Remarks

Link using `-lstdc++fs` to use these types and functions.

### 3.15.2.2 Typedef Documentation

#### file\_time\_type

using `std::experimental::filesystem::v1::file_time_type`

The type used for file timestamps.

### 3.15.2.3 Enumeration Type Documentation

#### copy\_options

enum class `std::experimental::filesystem::v1::copy_options` : unsigned short [strong]

Bitmask type controlling effects of `filesystem::copy`.

#### directory\_options

enum class `std::experimental::filesystem::v1::directory_options` : unsigned char [strong]

Bitmask type controlling directory iteration.

#### file\_type

enum class `std::experimental::filesystem::v1::file_type` : signed char [strong]

Enumerated type representing the type of a file.

#### perms

enum class `std::experimental::filesystem::v1::perms` : unsigned [strong]

Bitmask type representing file access permissions.

### 3.15.3 Library Fundamentals TS

Collaboration diagram for Library Fundamentals TS:



#### Topics

- [Array creation functions](#)
- [Const-propagating wrapper](#)
- [Detection idiom](#)
- [Logical operator traits](#)

- [Optional values](#)
- [Type-safe container of any type](#)
- [Variable template for type traits](#)

## Files

- file [algorithm](#)
- file [any](#)
- file [array](#)
- file [chrono](#)
- file [deque](#)
- file [forward\\_list](#)
- file [functional](#)
- file [iterator](#)
- file [list](#)
- file [map](#)
- file [memory](#)
- file [memory\\_resource](#)
- file [numeric](#)
- file [optional](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [string](#)
- file [string\\_view](#)
- file [system\\_error](#)
- file [tuple](#)
- file [type\\_traits](#)
- file [unordered\\_map](#)
- file [unordered\\_set](#)
- file [utility](#)
- file [vector](#)
- file [propagate\\_const](#)

## Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view<\\_CharT, \\_Traits>](#)

### 3.15.3.1 Detailed Description

Components defined by the *C++ Extensions for Library Fundamentals* Technical Specification, versions 1 and 2.

- ISO/IEC TS 19568:2015 C++ Extensions for Library Fundamentals
- ISO/IEC TS 19568:2017 C++ Extensions for Library Fundamentals, Version 2

### 3.15.3.2 Array creation functions

Collaboration diagram for Array creation functions:



#### Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::\_\_to\_array (_Tp(&__a)[_Nm]↔`  
`, index\_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>`  
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make\_array`  
`(_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to\_array (_Tp(&__a)[_Nm]) noexcept(is\_nothrow\_constructible<`  
`remove_cv_t< _Tp >, _Tp & ::value)`

#### 3.15.3.2.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

#### 3.15.3.2.2 Function Documentation

##### `make_array()`

```

template<typename _Dest = void, typename... _Types>
array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::↔
fundamentals_v2::make_array (
 _Types &&... __t) [constexpr]

```

Create a `std::array` from a variable-length list of arguments.

References [std::forward\(\)](#).

##### `to_array()`

```

template<typename _Tp, size_t _Nm>
array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (
 _Tp(&) __a[_Nm]) [constexpr], [noexcept]

```

Create a `std::array` from an array.



### 3.15.3.3 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



#### Classes

- class `std::experimental::fundamentals_v2::propagate_const<_Tp>`

#### Functions

- `template<typename _Tp>`  
`constexpr const _Tp & std::experimental::get_underlying (const propagate_const<_Tp> &__pt) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp & std::experimental::get_underlying (propagate_const<_Tp> &__pt) noexcept`
- `constexpr std::experimental::fundamentals_v2::__propagate_const_conversion_nc<_Tp, _Elem, true>::operator _Elem * ()`
- `constexpr std::experimental::fundamentals_v2::__propagate_const_conversions<_Tp * >::operator _Tp * () noexcept`
- `constexpr std::experimental::fundamentals_v2::__propagate_const_conversion_c<_Tp, _Elem, true>::operator const _Elem * () const`
- `constexpr std::experimental::fundamentals_v2::__propagate_const_conversions<_Tp * >::operator const _Tp * () const noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const _Tp &__t, const propagate_const<_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate_const<_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate_const<_Tp> &__pt, const propagate_const<_Up> &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (const propagate_const<_Tp> &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (nullptr_t, const propagate_const<_Tp> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const _Tp &__t, const propagate_const<_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate_const<_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate_const<_Tp> &__pt, const propagate_const<_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const _Tp &__t, const propagate_const<_Up> &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate_const<_Tp> &__pt, const _Up &__u)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (nullptr_t, const propagate\_const< _Tp > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > std::experimental::swap (propagate\_const< _Tp > &__pt, propagate\_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

### 3.15.3.3.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".

### 3.15.3.4 Detection idiom

Collaboration diagram for Detection idiom:



- `#define \_\_cpp\_lib\_experimental\_detect`

- `template<typename...>`  
using `std::experimental::void_t`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
using `std::experimental::detected_or`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
using `std::experimental::detected_or_t`
- `template<template< typename... > class _Op, typename... _Args>`  
using `std::experimental::detected_t`
- `template<template< typename... > class _Op, typename... _Args>`  
using `std::experimental::is_detected`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
using `std::experimental::is_detected_exact`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
using `std::experimental::is_detected_convertible`
- `template<template< typename... > class _Op, typename... _Args>`  
constexpr bool `std::experimental::is_detected_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
constexpr bool `std::experimental::is_detected_exact_v`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
constexpr bool `std::experimental::is_detected_convertible_v`

#### 3.15.3.4.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

#### 3.15.3.4.2 Macro Definition Documentation

##### `__cpp_lib_experimental_detect`

```
#define __cpp_lib_experimental_detect
```

A metafunction that always yields void, used for detecting valid types.

#### 3.15.3.4.3 Typedef Documentation

##### `detected_or`

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or
```

A metafunction that always yields void, used for detecting valid types.

##### `detected_or_t`

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or_t
```

A metafunction that always yields void, used for detecting valid types.

##### `detected_t`

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_t
```

A metafunction that always yields void, used for detecting valid types.

**is\_detected**

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected
```

A metafunction that always yields void, used for detecting valid types.

**is\_detected\_convertible**

```
template<typename _To, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_convertible
```

A metafunction that always yields void, used for detecting valid types.

**is\_detected\_exact**

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_exact
```

A metafunction that always yields void, used for detecting valid types.

**void\_t**

```
template<typename...>
using std::experimental::fundamentals_v2::void_t
```

A metafunction that always yields void, used for detecting valid types.

**3.15.3.4.4 Variable Documentation****is\_detected\_convertible\_v**

```
template<typename _To, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_convertible_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

**is\_detected\_exact\_v**

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_exact_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

**is\_detected\_v**

```
template<template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

### 3.15.3.5 Logical operator traits

Collaboration diagram for Logical operator traits:



- `#define __cpp_lib_experimental_logical_traits`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::conjunction_v`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::disjunction_v`
- `template<typename _Pp>`  
`constexpr bool std::experimental::negation_v`

#### 3.15.3.5.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

### 3.15.3.6 Optional values

Collaboration diagram for Optional values:



#### Classes

- class `std::experimental::fundamentals_v1::bad_optional_access`
- struct `std::experimental::fundamentals_v1::in_place_t`
- struct `std::experimental::fundamentals_v1::nullopt_t`
- class `std::experimental::fundamentals_v1::optional<_Tp>`

#### Macros

- `#define __cpp_lib_experimental_optional`

#### Variables

- `constexpr in_place_t std::experimental::in_place`
- `constexpr nullopt_t std::experimental::nullopt`

### 3.15.3.6.1 Detailed Description

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

### 3.15.3.6.2 Variable Documentation

#### in\_place

`in_place_t` `std::experimental::fundamentals_v1::in_place` [constexpr]

Tag for in-place construction.

#### nullopt

`nullopt_t` `std::experimental::fundamentals_v1::nullopt` [constexpr]

Tag to disengage optional objects.

### 3.15.3.7 Type-safe container of any type

Collaboration diagram for Type-safe container of any type:



### Classes

- class `std::experimental::fundamentals_v1::any`
- class `std::experimental::fundamentals_v1::bad_any_cast`

### Macros

- `#define __cpp_lib_experimental_any`

### Functions

- static void `std::experimental::fundamentals_v1::any::_Manager_external< _Tp >::_S_manage` (`_Op __` ← which, const `any` \*`__anyp`, `_Arg` \*`__arg`)
- static void `std::experimental::fundamentals_v1::any::_Manager_internal< _Tp >::_S_manage` (`_Op __` ← which, const `any` \*`__anyp`, `_Arg` \*`__arg`)
- template<typename `_ValueType`>  
`_ValueType` `std::experimental::any_cast` (const `any` &`__any`)
- void `std::experimental::swap` (`any` &`__x`, `any` &`__y`) noexcept
- template<typename `_ValueType`>  
`_ValueType` `std::experimental::any_cast` (`any` &`__any`)
- template<typename `_ValueType`, typename `enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>`>  
`_ValueType` `std::experimental::any_cast` (`any` &&`__any`)

- `template<typename _ValueType>`  
`const _ValueType * std::experimental::any\_cast (const any *__any) noexcept`
- `template<typename _ValueType>`  
`_ValueType * std::experimental::any\_cast (any *__any) noexcept`

### 3.15.3.7.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

### 3.15.3.7.2 Function Documentation

#### **`any_cast()`** [1/5]

```
template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference
_ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
 any && __any) [inline]
```

Access the contained object.

#### Template Parameters

|                         |                                        |
|-------------------------|----------------------------------------|
| <code>_ValueType</code> | A reference or CopyConstructible type. |
|-------------------------|----------------------------------------|

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__any</code> | The object to access. |
|--------------------|-----------------------|

#### Returns

The contained object.

#### Exceptions

|                                           |                                                                              |
|-------------------------------------------|------------------------------------------------------------------------------|
| <a href="#"><code>bad_any_cast</code></a> | If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code> |
|-------------------------------------------|------------------------------------------------------------------------------|

References [any\\_cast\(\)](#).

#### **`any_cast()`** [2/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
 any & __any) [inline]
```

Access the contained object.

#### Template Parameters

|                         |                                        |
|-------------------------|----------------------------------------|
| <code>_ValueType</code> | A reference or CopyConstructible type. |
|-------------------------|----------------------------------------|

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__any</code> | The object to access. |
|--------------------|-----------------------|

**Returns**

The contained object.

**Exceptions**

|                                           |                                                                              |
|-------------------------------------------|------------------------------------------------------------------------------|
| <code><a href="#">bad_any_cast</a></code> | If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code> |
|-------------------------------------------|------------------------------------------------------------------------------|

References [any\\_cast\(\)](#).

**any\_cast()** [3/5]

```
template<typename _ValueType>
_VValueType * std::experimental::fundamentals_v1::any_cast (
 any * __any) [inline], [noexcept]
```

Access the contained object.

**Template Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>_ValueType</code> | The type of the contained object. |
|-------------------------|-----------------------------------|

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__any</code> | A pointer to the object to access. |
|--------------------|------------------------------------|

**Returns**

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

**any\_cast()** [4/5]

```
template<typename _ValueType>
_VValueType std::experimental::fundamentals_v1::any_cast (
 const any & __any) [inline]
```

Access the contained object.

**Template Parameters**

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>_ValueType</code> | A const-reference or CopyConstructible type. |
|-------------------------|----------------------------------------------|



**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__any</code> | The object to access. |
|--------------------|-----------------------|

**Returns**

The contained object.

**Exceptions**

|                           |                                                                              |
|---------------------------|------------------------------------------------------------------------------|
| <code>bad_any_cast</code> | If <code>__any.type() != typeid(remove_reference_t&lt;_ValueType&gt;)</code> |
|---------------------------|------------------------------------------------------------------------------|

References [any\\_cast\(\)](#).

Referenced by [any\\_cast\(\)](#), [any\\_cast\(\)](#), and [any\\_cast\(\)](#).

**any\_cast()** [5/5]

```
template<typename _ValueType>
const _ValueType * std::experimental::fundamentals_v1::any_cast (
 const any * __any) [inline], [noexcept]
```

Access the contained object.

**Template Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>_ValueType</code> | The type of the contained object. |
|-------------------------|-----------------------------------|

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__any</code> | A pointer to the object to access. |
|--------------------|------------------------------------|

**Returns**

The address of the contained object if `__any != nullptr` && `__any.type() == typeid(↳ ValueType)` , otherwise a null pointer.

**swap()**

```
void std::experimental::fundamentals_v1::swap (
 any & __x,
 any & __y) [inline], [noexcept]
```

Exchange the states of two any objects.

**3.15.3.8 Variable template for type traits**

Collaboration diagram for Variable template for type traits:



- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_void_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_null_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_integral_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_floating_point_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_array_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_lvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_rvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_union_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_class_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_function_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_fundamental_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_object_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_compound_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_const_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_volatile_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivial_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pod_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_empty_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_final_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_signed_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_unsigned_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_constructible_v`

- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_trivially_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_trivially_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_nothrow_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::has_virtual_destructor_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::alignment_of_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::rank_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::extent_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_same_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_same_v< _Tp, _Tp >`

- `template<typename _Base, typename _Derived>`  
`constexpr bool std::experimental::is_base_of_v`
- `template<typename _From, typename _To>`  
`constexpr bool std::experimental::is_convertible_v`

#### 3.15.3.8.1 Detailed Description

Since

Library Fundamentals TS v1. C++14.

See also

`variable_templates`

### 3.15.4 Parallelism TS

Collaboration diagram for Parallelism TS:



#### Topics

- [Data parallel extensions](#)

#### 3.15.4.1 Detailed Description

Components defined by the *C++ Extensions for Parallelism* Technical Specification.

- ISO/IEC TS 19570:2015 C++ Extensions for Parallelism
- ISO/IEC TS 19570:2018 C++ Extensions for Parallelism, Version 2

#### 3.15.4.2 Data parallel extensions

Collaboration diagram for Data parallel extensions:



#### Macros

- `#define __cpp_lib_experimental_parallel_simd`

#### 3.15.4.2.1 Detailed Description

Data-parallel types library.

Since

C++17

### 3.16 Utilities

Collaboration diagram for Utilities:



#### Topics

- [Concepts](#)
- [Function Objects](#)
- [Memory](#)
- [Metaprogramming](#)
- [Rational Arithmetic](#)
- [Time](#)

#### Classes

- class [std::bitset<\\_Nb>](#)
- struct [std::pair<\\_T1, \\_T2>](#)
- struct [std::piecewise\\_construct\\_t](#)
- class [std::tuple<\\_Elements>](#)
- class [std::tuple<\\_T1, \\_T2>](#)

- struct `std::tuple_element< __i, tuple< _Types... > >`
- struct `std::tuple_size< tuple< _Elements... > >`
- struct `std::type_index`
- struct `std::uses_allocator< tuple< _Types... >, _Alloc >`

## Typedefs

- template<typename \_Tp, typename \_Up>  
using `std::__like_t`

## Functions

- template<typename... \_Args1, typename... \_Args2>  
constexpr `std::pair< _T1, _T2 >::pair` (`piecewise_construct_t`, `tuple< _Args1... >`, `tuple< _Args2... >`)
- template<typename \_Tp>  
constexpr `_Tp * std::__addressof` (`_Tp &__r`) noexcept
- template<typename \_Tp, typename \_Up = \_Tp>  
constexpr `_Tp std::__exchange` (`_Tp &__obj`, `_Up &&__new_val`)
- template<size\_t \_\_i, typename \_Head, typename... \_Tail>  
constexpr `_Head & std::__get_helper` (`_Tuple_impl< __i, _Head, _Tail... > &__t`) noexcept
- template<size\_t \_\_i, typename \_Head, typename... \_Tail>  
constexpr const `_Head & std::__get_helper` (const `_Tuple_impl< __i, _Head, _Tail... > &__t`) noexcept
- template<size\_t \_\_i, typename... \_Types>  
`__enable_if_t<(__i >= sizeof...( _Types))> std::__get_helper` (const `tuple< _Types... > &`)=delete
- template<typename \_Tp, typename \_Up = typename \_\_inv\_unwrap<\_Tp>::type>  
constexpr `_Up && std::__invfwd` (typename `remove_reference< _Tp >::type &__t`) noexcept
- template<typename \_Callable, typename... \_Args>  
constexpr `__invoke_result< _Callable, _Args... >::type std::__invoke` (`_Callable &&__fn`, `_Args &&... __args`)  
noexcept(`__is_nothrow_invocable< _Callable, _Args... >::value`)
- template<typename \_Res, typename \_MemFun, typename \_Tp, typename... \_Args>  
constexpr `_Res std::__invoke_impl` (`__invoke_memfun_deref`, `_MemFun &&__f`, `_Tp &&__t`, `_Args &&... __args`)
- template<typename \_Res, typename \_MemFun, typename \_Tp, typename... \_Args>  
constexpr `_Res std::__invoke_impl` (`__invoke_memfun_ref`, `_MemFun &&__f`, `_Tp &&__t`, `_Args &&... __args`)
- template<typename \_Res, typename \_MemPtr, typename \_Tp>  
constexpr `_Res std::__invoke_impl` (`__invoke_memobj_deref`, `_MemPtr &&__f`, `_Tp &&__t`)
- template<typename \_Res, typename \_MemPtr, typename \_Tp>  
constexpr `_Res std::__invoke_impl` (`__invoke_memobj_ref`, `_MemPtr &&__f`, `_Tp &&__t`)
- template<typename \_Res, typename \_Fn, typename... \_Args>  
constexpr `_Res std::__invoke_impl` (`__invoke_other`, `_Fn &&__f`, `_Args &&... __args`)
- template<typename \_Res, typename \_Callable, typename... \_Args>  
constexpr `enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > std::__invoke_r` (`_Callable &&__fn`, `_Args &&... __args`) noexcept(`is_nothrow_invocable_r_v< _Res, _Callable, _Args... >`)
- template<typename \_Cat, typename \_Tp, typename \_Up, typename \_IndexSeq>  
constexpr `_Cat std::__tuple_cmp` (const `_Tp &__t`, const `_Up &__u`, `_IndexSeq __indices`)
- template<typename \_Tp>  
constexpr `_Tp * std::addressof` (`_Tp &__r`) noexcept
- template<typename \_Tp>  
const `_Tp * std::addressof` (const `_Tp &&`)=delete
- template<typename \_Tp>  
auto `std::declval` () noexcept -> `decltype(__declval< _Tp >())`
- template<typename \_Tp>  
constexpr `_Tp && std::forward` (typename `std::remove_reference< _Tp >::type &&__t`) noexcept

- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr __like_t< _Tp, _Up > std::forward_like (_Up &&__x) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<typename _T1, typename _T2>`  
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)`
- `template<typename _Tp>`  
`constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (_Tp &__x) noexcept`
- `template<typename... _Tps, typename... _Ups>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > std::operator<=> (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename... _Tps, typename... _Ups>`  
`constexpr bool std::operator== (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename _Tp>`  
`constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > std::tie (_Elements &&... __args) noexcept`
- `template<typename... _UTypes>`  
`std::tuple (_UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>`  
`std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>`  
`std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >`



- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto std::tuple\_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::__type`

## Variables

- `constexpr piecewise\_construct\_t std::piecewise\_construct`
- `template<typename... _Types>`  
`constexpr size_t std::tuple\_size\_v< const tuple<_Types...> >`
- `template<typename... _Types>`  
`constexpr size_t std::tuple\_size\_v< tuple<_Types...> >`
- `template<typename _T1, typename _T2>`  
`pair (_T1, _T2) -> pair<_T1, _T2>`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t<__detail::__synth3way_t<_T1, _U1>, __detail::__synth3way_t<_T2, _U2>> operator<=> (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if<__and<__is_swappable<_T1>, __is_swappable<_T2>>::value>::type swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if<__and<__is_swappable<_T1>, __is_swappable<_T2>>::value>::type swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t<__detail::__synth3way_t<_T1, _U1>, __detail::__synth3way_t<_T2, _U2>> operator<=> (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`

### 3.16.1 Detailed Description

Basic function and class templates used with the rest of the library. Includes pair, swap, forward/move helpers, declval, integer\_sequence.

### 3.16.2 Function Documentation

#### [pair\(\)](#) [1/2]

```
template<typename _T1, typename _T2>
pair (
 _T1 ,
 _T2) -> pair<_T1, _T2> [related]
```

Two pairs are equal iff their members are equal.

#### [pair\(\)](#) [2/2]

```
template<class _T1, class _T2>
template<typename... _Args1, typename... _Args2>
std::pair<_T1, _T2>::pair (
 piecewise_construct_t ,
 tuple<_Args1...> __first,
 tuple<_Args2...> __second) [inline], [constexpr]
```

"piecewise construction" using a tuple of arguments for each member.

**Parameters**

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__first</code>  | Arguments for the first member of the pair.  |
| <code>__second</code> | Arguments for the second member of the pair. |

The elements of each tuple will be used as the constructor arguments for the data members of the pair.

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

**`__addressof()`**

```
template<typename _Tp>
_Tp * std::__addressof (
 _Tp & __r) [inline], [constexpr], [noexcept]
```

Same as C++11 `std::addressof`.

Referenced by [std::back\\_insert\\_iterator< \\_Container >::back\\_insert\\_iterator\(\)](#), [std::basic\\_istream< char >::basic\\_istream\(\)](#), [std::basic\\_istream< char >::basic\\_istream\(\)](#), [std::basic\\_istream< char >::basic\\_istream\(\)](#), [std::basic\\_ostream< char >::basic\\_ostream\(\)](#), [std::basic\\_ostream< char >::basic\\_ostream\(\)](#), [std::basic\\_ostream< char >::basic\\_ostream\(\)](#), [std::basic\\_stringstream< char >::basic\\_stringstream\(\)](#), [std::basic\\_stringstream< char >::basic\\_stringstream\(\)](#), [std::front\\_insert\\_iterator< \\_Container >::front\\_insert\\_iterator\(\)](#), [std::insert\\_iterator< \\_Container >::insert\\_iterator\(\)](#), [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >::istream\\_iterator\(\)](#), [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >::ostream\\_iterator\(\)](#), [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >::ostream\\_iterator\(\)](#), [\\_\\_gnu\\_debug:: Safe\\_sequence< \\_Sequence >:: M\\_transfer\\_from\\_if](#), [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::forward\\_list< \\_Tp, \\_Alloc >::merge\(\)](#), [std::list< \\_Tp, \\_Alloc >::merge\(\)](#), [std::list< \\_Tp, \\_Alloc >::merge\(\)](#), [std::sub\\_match< const char \\* >::operator<<\(\)](#), [\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, map >](#), [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#), [std::basic\\_string< \\_CharT >::operator=\(\)](#), [std::deque< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::forward\\_list< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::list< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::vector< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::sub\\_match< const char \\* >::operator==\(\)](#), [std::basic\\_istream< char >::rdbuf\(\)](#), [std::basic\\_ostream< char >::rdbuf\(\)](#), [std::basic\\_stringstream< char >::rdbuf\(\)](#), [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::splice\(\)](#) and [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::splice\(\)](#).

**`__invoke()`**

```
template<typename _Callable, typename... _Args>
__invoke_result< _Callable, _Args... >::type std::__invoke (
 _Callable && __fn,
 _Args &&... __args) [constexpr], [noexcept]
```

Invoke a callable object.

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

Referenced by [std::function\\_ref< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::function\\_ref\(\)](#), and [std::once\\_flag::call\\_once](#).

**`addressof()`**

```
template<typename _Tp>
_Tp * std::addressof (
 _Tp & __r) [inline], [constexpr], [noexcept]
```

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

**Parameters**

|                                                                                   |                                     |
|-----------------------------------------------------------------------------------|-------------------------------------|
|  | Reference to an object or function. |
|  |                                     |
|  |                                     |
|  |                                     |
| <i>r</i>                                                                          |                                     |

**Returns**

The actual address.

**Since**

C++11

Referenced by [std::function\\_ref< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::function\\_ref\(\)](#), [std::function\\_ref< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::function\\_ref\(\)](#), [\\_Destroy\(\)](#), [\\_Destroy\\_n\(\)](#), [std::copyable\\_function< \\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>::operator=\(\)](#), and [std::move\\_only\\_function< \\_Res](#)

**declval()**

```
template<typename _Tp>
auto std::declval () -> decltype(__declval< _Tp >(0)) [noexcept]
```

Utility to simplify expressions used in unevaluated operands

**Since**

C++11

Referenced by [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::vector\(\)](#), and [std::unique\\_ptr< \\_State >::operator\\*\(\)](#).

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

```
template<typename _Tp>
_Tp && std::forward (
 typename std::remove_reference< _Tp >::type && __t) [constexpr], [noexcept]
```

Forward an rvalue.

**Returns**

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

**Since**

C++11

**forward() [2/2]**

```
template<typename _Tp>
_Tp && std::forward (
 typename std::remove_reference< _Tp >::type & __t) [constexpr], [noexcept]
```

Forward an lvalue.

**Returns**

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

Referenced by `std::experimental::fundamentals_v1::any::any()`, `std::copyable_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept`, `std::copyable_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function()`, `std::copyable_function<_Res`, `std::function<_Res(_ArgTypes...)>::function()`, `std::function_ref<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function`, `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function()`, `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function()`, `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function()`, `std::pair<const __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>::pair()`, `std::unique_ptr<_State>::unique_ptr()`, `__invoke()`, `_Construct()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::M_create_node()`, `std::deque<_Tp, _Alloc>::M_push_back`, `std::deque<_Tp, _Alloc>::M_push_front_aux()`, `std::shared_ptr<_State_base>::allocate_shared()`, `async()`, `async()`, `bind()`, `std::once_flag::call_once`, `std::allocator_traits<_OuterAlloc>::construct()`, `std::allocator_traits<allocator<_Tp>>::con`, `std::allocator_traits<allocator<void>>::construct()`, `std::allocator_traits<pmr::polymorphic_allocator<_Tp>>::construct()`, `std::deque<_Tp, _Alloc>::emplace()`, `std::list<_Tp, _Alloc>::emplace()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const`, `std::multimap<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>::emplace()`, `std::multiset<_Key, _Cmp, polymo`, `std::set<_Key, _Cmp, polymorphic_allocator<_Key>>::emplace()`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allo`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::emplace()`, `std::unordered_multiset<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::emplace()`, `std::unordered_set<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::emplace()`, `std::vector<_Tp, polymorphic_allocator<_Tp>`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::emplace_after()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::em`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>::emplace_hint()`, `std::multimap<_Key, _Tp, _Cmp, p`, `std::multiset<_Key, _Cmp, polymorphic_allocator<_Key>>::emplace_hint()`, `std::set<_Key, _Cmp, polymorphic_allocator<_Key>>`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::emplace←`, `_hint()`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::emplace_hint()`, `std::unordered_multiset<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::emplace_hint()`, `std::unordered_set<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::emplace_hint()`, `std::pair<const __iter_key_t<_InputItera`, `std::pair<const __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>::explicit()`, `std::pair<const __iter_key_t<_InputItera`, `forward_as_tuple()`, `get()`, `get()`, `get()`, `get()`, `get()`, `get()`, `inout_ptr()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Ke`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>::insert()`, `std::multimap<_Key, _Tp, _Cmp, polymor`, `std::multimap<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>::insert()`, `std::unordered_map<_Key, _Tp, _Ha`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::insert()`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::insert()`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>::insert()`, `std::unordered_set<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::insert()`, `std::unordered_set<_Key, _Hash, _Pred, polymo`, `std::experimental::make_array()`, `std::experimental::make_ostream_joiner()`, `make_pair()`, `std::shared_ptr<_State_base>::make_shared`, `make_tuple()`, `std::experimental::not_fn()`, `std::copyable_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator`, `std::function<_Res(_ArgTypes...)>::operator()`, `std::function_ref<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::opera`, `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator()`, `std::copyable_function<_Res(_Ar`, `std::experimental::fundamentals_v1::any::operator=()`, `std::function<_Res(_ArgTypes...)>::operator=()`, `std::move_only_function<_Res(_`, `std::pair<const __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>::operator=()`, `std::pair<const __iter_key_t<_InputItera`, `std::pair<const __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>::operator=()`, `std::pair<const __iter_key_t<_InputItera`, `std::unique_ptr<_State>::operator=()`, `std::unique_ptr<_Tp[], _Dp>::operator=()`, `operator>>()`, `std::unordered_map<_Key, _Tp, _H`, `out_ptr()`, `std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>>::reserve()`, `std::experimental::sample()`, `sample()`, `throw_with_nested()`, and `tuple_cat()`.

**forward\_as\_tuple()**

```
template<typename... _Elements>
tuple<_Elements &&...> std::forward_as_tuple (
 _Elements &&... __args) [constexpr, [noexcept]
```

Create a tuple of lvalue or rvalue references to the arguments.  
References [forward\(\)](#).

### forward\_like()

```
template<typename _Tp, typename _Up>
__like_t< _Tp, _Up > std::forward_like (
 _Up && __x) [nodiscard], [constexpr], [noexcept]
```

Forward with the cv-qualifiers and value category of another type.

### Template Parameters

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <code>_Tp</code> | An lvalue reference or rvalue reference.                     |
| <code>_Up</code> | An lvalue reference type deduced from the function argument. |

### Parameters

|                  |            |
|------------------|------------|
| <code>__x</code> | An lvalue. |
|------------------|------------|

### Returns

`__x` converted to match the qualifiers of `_Tp`.

### Since

C++23

### get() [1/4]

```
template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > && std::get (
 const tuple< _Elements... > && __t) [constexpr], [noexcept]
```

Return a const rvalue reference to the *i*th element of a const tuple rvalue.

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

### get() [2/4]

```
template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > & std::get (
 const tuple< _Elements... > & __t) [constexpr], [noexcept]
```

Return a const reference to the *i*th element of a const tuple.

### get() [3/4]

```
template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > && std::get (
 tuple< _Elements... > && __t) [constexpr], [noexcept]
```

Return an rvalue reference to the *i*th element of a tuple rvalue.

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

**get()** [4/4]

```
template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > & std::get (
 tuple< _Elements... > & __t) [constexpr], [noexcept]
```

Return a reference to the *i*th element of a tuple.

**make\_pair()**

```
template<typename _T1, typename _T2>
pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >
std::make_pair (
 _T1 && __x,
 _T2 && __y) [constexpr]
```

A convenience wrapper for creating a pair from two objects.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| $\leftarrow$<br>_x | The first object.  |
| $\leftarrow$<br>_y | The second object. |

**Returns**

A newly-constructed pair<> object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).

Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.

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

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gen\\_two\\_uniform\\_ints\(\)](#), [\\_\\_gnu\\_debug::\\_\\_get\\_distance\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_merge\\_advance\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector::\\_\\_M\\_sequential\\_algorithm\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [\\_\\_gnu\\_pbds::detail::pat\\_citer\\_base::\\_\\_Node\\_citer<node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc>::valid\\_prefix\(\)](#).

**make\_tuple()**

```
template<typename... _Elements>
tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (
 _Elements &&... __args) [constexpr]
```

Create a tuple containing copies of the arguments.

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

**move()**

```
template<typename _Tp>
std::remove_reference< _Tp >::type && std::move (
 _Tp && __t) [constexpr], [noexcept]
```

Convert a value to an rvalue.

**Parameters**

|          |                            |
|----------|----------------------------|
| ↔        | A thing of arbitrary type. |
| ↔        |                            |
| ↔        |                            |
| ↔        |                            |
| <i>t</i> |                            |

**Returns**

The parameter cast to an rvalue-reference to allow moving it.

**Since**

C++11

Referenced by `__gnu_cxx::__versa_string<char>::__versa_string()`, `std::basic_string<CharT>::basic_string()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::deque()`, `std::discard_block_engine<ranlux24_base, 223, 23>::discard_block_eng`, `std::tr2::dynamic_bitset<_WordT, _Alloc>::dynamic_bitset()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::forward_list()`, `std::future<_Res>::future()`, `std::future<_Res &>::future()`, `std::future<void>::future()`, `std::independent_bits_engine<_RandomNum`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::map()`, `std::match_results<↔`, `_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::↔`, `match_results()`, `std::multimap<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::multimap()`, `std::multiset<_Key, _Cmp, polymorphic_allocator<_Key>>::multiset()`, `std::priority_queue<_Tp, _Sequence, _Compare>::priority_q`, `std::set<_Key, _Cmp, polymorphic_allocator<_Key>>::set()`, `std::shared_future<_Res>::shared_future()`, `std::shared_future<_Res>::shared_future()`, `std::shared_future<_Res &>::shared_future()`, `std::shared_future<_Res &>::shared_fu`, `std::shared_future<void>::shared_future()`, `std::shared_future<void>::shared_future()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shared_ptr<_State_base>::shared_ptr()`, `std::shuffle_order_engine<minstd_rand0, 256>::shuffle_order_engine()`, `std::unique_ptr<_State>::unique_ptr()`, `std::unique_ptr<_Tp[], _Dp>::unique_ptr()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::vector()`, `std::unique_ptr<_State>::~`, `std::experimental::any_cast()`, `std::basic_regex<char>::assign()`, `std::basic_string<CharT>::assign()`, `async()`, `atomic_compare_exchange_strong()`, `atomic_compare_exchange_strong()`, `atomic_compare_exchange_strong_explicit()`, `atomic_compare_exchange_strong_explicit()`, `atomic_compare_exchange_weak()`, `atomic_compare_exchange_weak()`, `atomic_compare_exchange_weak_explicit()`, `atomic_compare_exchange_weak_explicit()`, `atomic_exchange()`, `atomic_exchange()`, `atomic_store()`, `atomic_store()`, `std::shared_ptr<_State_base>::const_pointer_cast()`, `copy_n()`, `distance()`, `std::shared_ptr<_State_base>::dynamic_pointer_cast()`, `exclusive_scan()`, `exclusive_scan()`, `for_each_n()`, `std::future<_Res>::get()`, `get()`, `get()`, `inclusive_scan()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::multimap<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::multimap<_Key, _Tp, _Cmp, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::set<_Key, _Cmp, polymorphic_a`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>>::insert()`, `std::unordered_multiset<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::insert()`, `std::unordered_multiset<_Key, _Hash, _Pre`, `std::unordered_set<_Key, _Hash, _Pred, polymorphic_allocator<_Key>>::insert()`, `std::unordered_set<_Key, _Hash, _Pred, polymo`, `std::vector<_Tp, _Alloc>::insert()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::insert()`, `std::experimental::make_boyer_moore`, `std::experimental::make_boyer_moore_searcher()`, `std::forward_list<_Tp, _Alloc>::merge()`, `std::forward_list<_Tp, polymorphic_allocat`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::merge()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::merge()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::merge()`, `move_if_noexcept()`, `std::tr2::dynamic_bitset<_WordT, _Alloc>::operator&=(`, `std::chrono::operator<<()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::operator=()`, `std::forward_list<_Tp, polymorphic_allocat`, `std::function<_Res(_ArgTypes...)>::operator=()`, `std::list<_Tp, polymorphic_allocator<_Tp>>::operator=()`, `std::vector<_Tp, polymorphic_allocator<_Tp>>::operator=()`, `std::filesystem::path::operator>>`, `operator>>()`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, polymorphic_allocator<pair<const _Key, _Tp>>>::operator[]()`,

`reduce()`, `std::shared_ptr<_State_base>::reinterpret_pointer_cast()`, `std::unique_ptr<_State>::reset()`, `std::unique_ptr<_Tp[],_Dp>::reset()`, `std::list<_Tp,polymorphic_allocator<_Tp>>::splice()`, `std::list<_Tp,polymorphic_allocator<_Tp>>::splice()`, `std::forward_list<_Tp,polymorphic_allocator<_Tp>>::splice_after()`, `std::shared_ptr<_State_base>::static_pointer_cast()`, `transform_exclusive_scan()`, `transform_inclusive_scan()`, and `transform_reduce()`.

### **move\_if\_noexcept()**

```
template<typename _Tp>
__conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (
 _Tp & __x) [constexpr], [noexcept]
```

Conditionally convert a value to an rvalue.

### **Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__x</code> | A thing of arbitrary type. |
|------------------|----------------------------|

### **Returns**

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

### **Since**

C++11

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

### **operator<==>() [1/2]**

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<==> (
 const pair< _T1, _T2 > & __x,
 const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

### **operator<==>() [2/2]**

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<==> (
 const pair< _T1, _T2 > & __x,
 const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.



**operator==()** [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
 const pair< _T1, _T2 > & __x,
 const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

**operator==()** [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
 const pair< _T1, _T2 > & __x,
 const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

**swap()** [1/5]

```
template<typename _T1, typename _T2>
enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
 pair< _T1, _T2 > & __x,
 pair< _T1, _T2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

**swap()** [2/5]

```
template<typename _T1, typename _T2>
enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
 pair< _T1, _T2 > & __x,
 pair< _T1, _T2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

**swap()** [3/5]

```
template<typename _Tp>
void std::swap (
 _Tp & __a,
 _Tp & __b) [inline], [constexpr], [noexcept]
```

Swaps two values.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__a</code> | A thing of arbitrary type. |
|------------------|----------------------------|

|                        |                                  |
|------------------------|----------------------------------|
| <code>_↔<br/>_b</code> | Another thing of arbitrary type. |
|------------------------|----------------------------------|

**swap() [4/5]**

```
template<typename _Tp, size_t _Nm>
void std::swap (
 _Tp(&) __a[_Nm],
 _Tp(&) __b[_Nm]) [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

**swap() [5/5]**

```
template<typename... _Elements>
enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (
 tuple< _Elements... > & __x,
 tuple< _Elements... > & __y) [inline], [constexpr], [delete], [noexcept]
```

Exchange the values of two tuples.

**tie()**

```
template<typename... _Elements>
tuple< _Elements &... > std::tie (
 _Elements &... __args) [constexpr], [noexcept]
```

Return a tuple of lvalue references bound to the arguments.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#).

**tuple\_cat()**

```
template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::↔
value>::type>
auto std::tuple_cat (
 _Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type [constexpr]
```

Create a tuple containing all elements from multiple tuple-like objects.

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

**3.16.3 Variable Documentation****piecewise\_construct**

```
piecewise_construct_t std::piecewise_construct [inline], [constexpr]
```

Tag for piecewise construction of `std::pair` objects.

Referenced by [std::map< \\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::operator\[\]\(\)](#).

### 3.16.4 Concepts

Collaboration diagram for Concepts:



#### Files

- file [concepts](#)
- file [ranges](#)

#### 3.16.4.1 Detailed Description

Concepts for checking type requirements.

### 3.16.5 Function Objects

Collaboration diagram for Function Objects:



#### Topics

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Function Object Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

#### Classes

- `struct std::binary_function< _Arg1, _Arg2, _Result >`
- `class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- `class std::function< _Res(_ArgTypes...)>`
- `class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

- class `std::move_only_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `std::reference_wrapper<_Tp>`
- struct `std::unary_function<_Arg, _Result>`

## Functions

- template<typename \_Tp, typename \_Class>  
constexpr `_Mem_fn<_Tp _Class::*> std::mem_fn (_Tp _Class::* __pm)` noexcept

### 3.16.5.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables. Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing double, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will usually be inlined directly.

An *adaptable function object* is one which provides nested typedefs `result_type` and either `argument_type` (for a unary function) or `first_argument_type` and `second_argument_type` (for a binary function). Those typedefs are used by function object adaptors such as `bind2nd`. The standard library provides two class templates, `unary_function` and `binary_function`, which define those typedefs and so can be used as base classes of adaptable function objects.

Since C++11 the use of function object adaptors has been superseded by more powerful tools such as lambda expressions, `function<>`, and more powerful type deduction (using `auto` and `decltype`). The helpers for defining adaptable function objects are deprecated since C++11, and no longer part of the standard library since C++17. However, they are still defined and used by libstdc++ after C++17, as a conforming extension.

### 3.16.5.2 Function Documentation

#### `mem_fn()`

```
template<typename _Tp, typename _Class>
_Mem_fn<_Tp _Class::*> std::mem_fn (
 _Tp _Class::* __pm) [inline], [constexpr], [noexcept]
```

Returns a function object that forwards to the member pointer pointer `pm`.

This allows a pointer-to-member to be transformed into a function object that can be called with an object expression as its first argument.

For a pointer-to-data-member the result must be called with exactly one argument, the object expression that would be used as the first operand in a `obj.*memptr` or `objp->*memptr` expression.

For a pointer-to-member-function the result must be called with an object expression and any additional arguments to pass to the member function, as in an expression like `(obj.*memfun)(args...)` or `(objp->*memfun)(args...)`.

The object expression can be a pointer, reference, `reference_wrapper`, or smart pointer, and the call wrapper will dereference it as needed to apply the pointer-to-member.

Since

C++11

### 3.16.5.3 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



#### Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function< _Arg, _Result >`

#### Functions

- template<typename \_Arg, typename \_Result>  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun ( _Result(* __x)( _Arg))`
- template<typename \_Arg1, typename \_Arg2, typename \_Result>  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun ( _Result(* __x)( _Arg1, _Arg2))`

#### 3.16.5.3.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

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

#### 3.16.5.3.2 Function Documentation

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

```

template<typename _Arg, typename _Result>
pointer_to_unary_function< _Arg, _Result > std::ptr_fun (
 _Result(* __x) (_Arg)) [inline]

```

One of the [adaptors for function pointers](#).

References [ptr\\_fun\(\)](#).

Referenced by [ptr\\_fun\(\)](#), and [ptr\\_fun\(\)](#).

**ptr\_fun()** [2/2]

```
template<typename _Arg1, typename _Arg2, typename _Result>
pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (
 _Result (* __x) (_Arg1, _Arg2)) [inline]
```

One of the [adaptors for function pointers](#).

References [ptr\\_fun\(\)](#).

**3.16.5.4 Adaptors for pointers to members**

Collaboration diagram for Adaptors for pointers to members:

**Classes**

- class [std::const\\_mem\\_fun1\\_ref\\_t](#)< \_Ret, \_Tp, \_Arg >
- class [std::const\\_mem\\_fun1\\_t](#)< \_Ret, \_Tp, \_Arg >
- class [std::const\\_mem\\_fun\\_ref\\_t](#)< \_Ret, \_Tp >
- class [std::const\\_mem\\_fun\\_t](#)< \_Ret, \_Tp >
- class [std::mem\\_fun1\\_ref\\_t](#)< \_Ret, \_Tp, \_Arg >
- class [std::mem\\_fun1\\_t](#)< \_Ret, \_Tp, \_Arg >
- class [std::mem\\_fun\\_ref\\_t](#)< \_Ret, \_Tp >
- class [std::mem\\_fun\\_t](#)< \_Ret, \_Tp >

**Functions**

- template<typename \_Ret, typename \_Tp>  
[mem\\_fun\\_t](#)< \_Ret, \_Tp > **std::mem\_fun** (\_Ret(\_Tp::\*\_\_f)())
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
[mem\\_fun1\\_t](#)< \_Ret, \_Tp, \_Arg > **std::mem\_fun** (\_Ret(\_Tp::\*\_\_f)(\_Arg))
- template<typename \_Ret, typename \_Tp>  
[mem\\_fun\\_ref\\_t](#)< \_Ret, \_Tp > **std::mem\_fun\_ref** (\_Ret(\_Tp::\*\_\_f)())
- template<typename \_Ret, typename \_Tp, typename \_Arg>  
[mem\\_fun1\\_ref\\_t](#)< \_Ret, \_Tp, \_Arg > **std::mem\_fun\_ref** (\_Ret(\_Tp::\*\_\_f)(\_Arg))

**3.16.5.4.1 Detailed Description**

There are a total of  $8 = 2^3$  function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

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

### 3.16.5.5 Arithmetic Function Object Classes

Collaboration diagram for Arithmetic Function Object Classes:



#### Classes

- struct [std::divides< \\_Tp >](#)
- struct [std::divides< void >](#)
- struct [std::minus< \\_Tp >](#)
- struct [std::minus< void >](#)
- struct [std::modulus< \\_Tp >](#)
- struct [std::modulus< void >](#)
- struct [std::multiplies< \\_Tp >](#)
- struct [std::multiplies< void >](#)
- struct [std::negate< \\_Tp >](#)
- struct [std::negate< void >](#)
- struct [std::plus< \\_Tp >](#)

#### 3.16.5.5.1 Detailed Description

The library provides function objects for basic arithmetic operations. See the documentation for [function objects](#) for examples of their use.

### 3.16.5.6 Binder Classes

Collaboration diagram for Binder Classes:



#### Namespaces

- namespace [std::placeholders](#)



## Classes

- struct `std::_Placeholder<_Num>`
- class `std::binder1st<_Operation>`
- class `std::binder2nd<_Operation>`
- struct `std::is_bind_expression<_Tp>`
- struct `std::is_bind_expression<_Bind<_Signature>>`
- struct `std::is_bind_expression<_Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const _Bind<_Signature>>`
- struct `std::is_bind_expression<const _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<const volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_placeholder<_Tp>`
- struct `std::is_placeholder<_Placeholder<_Num>>`

## Functions

- template<typename \_Result, typename \_Func, typename... \_BoundArgs>  
constexpr `_Bindres_helper<_Result, _Func, _BoundArgs...>::type` `std::bind` (`_Func` &&\_\_f, `_BoundArgs` &&... \_\_args)
- template<typename \_Func, typename... \_BoundArgs>  
constexpr `_Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type` `std::bind` (`_Func` &&\_\_f, `_BoundArgs` &&... \_\_args)
- template<typename \_Operation, typename \_Tp>  
`binder1st<_Operation>` `std::bind1st` (const `_Operation` &\_\_fn, const `_Tp` &\_\_x)
- template<typename \_Operation, typename \_Tp>  
`binder2nd<_Operation>` `std::bind2nd` (const `_Operation` &\_\_fn, const `_Tp` &\_\_x)

### 3.16.5.6.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B`'s `operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(), 5)`).

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

### 3.16.5.6.2 Function Documentation

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

```
template<typename _Result, typename _Func, typename... _BoundArgs>
_Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (
 _Func && __f,
 _BoundArgs &&... __args) [inline], [constexpr]
```

Function template for `std::bind<R>`.

Since

C++11

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

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

```
template<typename _Func, typename... _BoundArgs>
_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (
 _Func && __f,
 _BoundArgs &&... __args) [inline], [constexpr]
```

Function template for `std::bind`.

Since

C++11

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

#### **bind1st()**

```
template<typename _Operation, typename _Tp>
binder1st< _Operation > std::bind1st (
 const _Operation & __fn,
 const _Tp & __x) [inline]
```

One of the [binder functions](#).

References [bind1st\(\)](#).

Referenced by [bind1st\(\)](#).

#### **bind2nd()**

```
template<typename _Operation, typename _Tp>
binder2nd< _Operation > std::bind2nd (
 const _Operation & __fn,
 const _Tp & __x) [inline]
```

One of the [binder functions](#).

References [bind2nd\(\)](#).

Referenced by [bind2nd\(\)](#).

### 3.16.5.7 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



#### Classes

- struct `std::logical_and< _Tp >`
- struct `std::logical_and< void >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`

#### 3.16.5.7.1 Detailed Description

The library provides function objects for the logical operations: `&&`, `||`, and `!`.

### 3.16.5.8 Comparison Classes

Collaboration diagram for Comparison Classes:



#### Classes

- struct `std::equal_to< _Tp >`
- struct `std::equal_to< void >`
- struct `std::greater< _Tp >`
- struct `std::greater< void >`
- struct `std::greater_equal< _Tp >`
- struct `std::greater_equal< void >`
- struct `std::less< _Tp >`
- struct `std::less< void >`
- struct `std::less_equal< _Tp >`

- struct `std::less_equal< void >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< void >`

#### 3.16.5.8.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

#### 3.16.5.9 Hashes

Collaboration diagram for Hashes:



#### Classes

- struct `std::hash< _Tp >`
- struct `std::hash< _Tp * >`
- struct `std::hash< bool >`
- struct `std::hash< char >`
- struct `std::hash< char16_t >`
- struct `std::hash< char32_t >`
- struct `std::hash< double >`
- struct `std::hash< float >`
- struct `std::hash< int >`
- struct `std::hash< long >`
- struct `std::hash< long double >`
- struct `std::hash< long long >`
- struct `std::hash< short >`
- struct `std::hash< signed char >`
- struct `std::hash< unsigned char >`
- struct `std::hash< unsigned int >`
- struct `std::hash< unsigned long >`
- struct `std::hash< unsigned long long >`
- struct `std::hash< unsigned short >`
- struct `std::hash< wchar_t >`

#### Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

## Variables

- `template<typename _Tp, typename = void>`  
`constexpr bool std::__is_hash_enabled_for`
- `template<typename _Tp>`  
`constexpr bool std::__is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp>{}(declval< _Tp>{}))>`  
`>`

### 3.16.5.9.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

### 3.16.5.10 Negators

Collaboration diagram for Negators:



## Classes

- class `std::binary_negate< _Predicate >`
- class `std::unary_negate< _Predicate >`

## Functions

- `template<typename _Predicate>`  
`constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`  
`constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`

### 3.16.5.10.1 Detailed Description

The function templates `not1` and `not2` are function object adaptors, which each take a predicate functor and wrap it in an instance of `unary_negate` or `binary_negate`, respectively. Those classes are functors whose `operator()` evaluates the wrapped predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
 bool operator() (int x) const { return x > 3; }
};

```

```
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates taking two arguments.

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

### 3.16.5.10.2 Function Documentation

#### not1()

```
template<typename _Predicate>
unary_negate< _Predicate > std::not1 (
 const _Predicate & __pred) [inline], [constexpr]
```

One of the [negation functors](#).

References [not1\(\)](#).

Referenced by [not1\(\)](#).

#### not2()

```
template<typename _Predicate>
binary_negate< _Predicate > std::not2 (
 const _Predicate & __pred) [inline], [constexpr]
```

One of the [negation functors](#).

References [not2\(\)](#).

Referenced by [not2\(\)](#).

## 3.16.6 Memory

Collaboration diagram for Memory:



### Topics

- [Allocators](#)
- [Pointer Abstractions](#)
- [Polymorphic memory resources](#)

### Files

- file [memory](#)

### Functions

- void \* [std::align](#) (size\_t \_\_align, size\_t \_\_size, void \*&\_\_ptr, size\_t &\_\_space) noexcept

- `template<size_t _Align, class _Tp>`  
`constexpr _Tp * std::assume\_aligned (_Tp * __ptr) noexcept`
- `template<size_t _Align, class _Tp>`  
`bool std::is\_sufficiently\_aligned (_Tp * __ptr)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_copy\_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void std::uninitialized\_default\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::uninitialized\_default\_construct\_n (_ForwardIterator __first, _Size __count)`
- `template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr void std::uninitialized\_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr _ForwardIterator std::uninitialized\_fill\_n (_ForwardIterator __first, _Size __n, const _Tp & __x)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr pair< _InputIterator, _ForwardIterator > std::uninitialized\_move\_n (_InputIterator __first, _Size __count, _ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void std::uninitialized\_value\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::uninitialized\_value\_construct\_n (_ForwardIterator __first, _Size __count)`

### 3.16.6.1 Detailed Description

Components for memory allocation, deallocation, and management.

### 3.16.6.2 Function Documentation

#### **align()**

```
void * std::align (
 size_t __align,
 size_t __size,
 void *& __ptr,
 size_t & __space) [inline], [noexcept]
```

Fit aligned storage in buffer.

This function tries to fit `__size` bytes of storage with alignment `__align` into the buffer `__ptr` of size `__space` bytes. If such a buffer fits then `__ptr` is changed to point to the first byte of the aligned storage and `__space` is reduced by the bytes used for alignment.

C++11 20.6.5 [ptr.align]

#### **Parameters**

|                      |                                                    |
|----------------------|----------------------------------------------------|
| <code>__align</code> | A fundamental or extended alignment value.         |
| <code>__size</code>  | Size of the aligned storage required.              |
| <code>__ptr</code>   | Pointer to a buffer of <code>__space</code> bytes. |

|                      |                                                       |
|----------------------|-------------------------------------------------------|
| <code>__space</code> | Size of the buffer pointed to by <code>__ptr</code> . |
|----------------------|-------------------------------------------------------|

#### Returns

the updated pointer if the aligned storage fits, otherwise `nullptr`.

#### **assume\_aligned()**

```
template<size_t _Align, class _Tp>
_Tp * std::assume_aligned (
 _Tp * __ptr) [nodiscard], [constexpr], [noexcept]
```

Inform the compiler that a pointer is aligned.

#### Template Parameters

|                     |                                          |
|---------------------|------------------------------------------|
| <code>_Align</code> | An alignment value (i.e. a power of two) |
| <code>_Tp</code>    | An object type                           |

#### Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__ptr</code> | A pointer that is aligned to <code>_Align</code> |
|--------------------|--------------------------------------------------|

C++20 20.10.6 [ptr.align]

#### **is\_sufficiently\_aligned()**

```
template<size_t _Align, class _Tp>
bool std::is_sufficiently_aligned (
 _Tp * __ptr) [inline], [nodiscard]
```

Is `__ptr` aligned to an `_Align` byte boundary?

#### Template Parameters

|                     |                    |
|---------------------|--------------------|
| <code>_Align</code> | An alignment value |
| <code>_Tp</code>    | An object type     |

C++26 20.2.5 [ptr.align]

#### **uninitialized\_copy()**

```
template<typename _InputIterator, typename _ForwardIterator>
_FowardIterator std::uninitialized_copy (
 _InputIterator __first,
 _InputIterator __last,
 _ForwardIterator __result) [inline], [constexpr]
```

Copies the range `[first,last)` into `result`.



**Parameters**

|                       |                     |
|-----------------------|---------------------|
| <code>__first</code>  | An input iterator.  |
| <code>__last</code>   | An input iterator.  |
| <code>__result</code> | A forward iterator. |

**Returns**

`__result + (__last - __first)`

Like `std::copy`, but does not require an initialized output range.

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

Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [uninitialized\\_move\(\)](#).

**uninitialized\_copy\_n()**

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
_FForwardIterator std::uninitialized_copy_n (
 _InputIterator __first,
 _Size __n,
 _ForwardIterator __result) [inline], [constexpr]
```

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

**Parameters**

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

**Returns**

`__result + __n`

**Since**

C++11

Like `copy_n()`, but does not require an initialized output range.

**uninitialized\_default\_construct()**

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

Default-initializes objects in the range `[first,last)`.

**Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__first</code> | A forward iterator. |
|----------------------|---------------------|

|                     |                     |
|---------------------|---------------------|
| <code>__last</code> | A forward iterator. |
|---------------------|---------------------|

Since

C++17

### **uninitialized\_default\_construct\_n()**

```
template<typename _ForwardIterator, typename _Size>
_FowardIterator std::uninitialized_default_construct_n (
 _ForwardIterator __first,
 _Size __count) [inline], [constexpr]
```

Default-initializes objects in the range [first,first+count).

#### **Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__first</code> | A forward iterator.                 |
| <code>__count</code> | The number of objects to construct. |

Returns

`__first + __count`

Since

C++17

### **uninitialized\_fill()**

```
template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
void std::uninitialized_fill (
 _ForwardIterator __first,
 _ForwardIterator __last,
 const _Tp & __x) [inline], [constexpr]
```

Copies the value x into the range [first,last).

#### **Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__first</code> | A forward iterator. |
| <code>__last</code>  | A forward iterator. |
| <code>__x</code>     | The source value.   |

Like `std::fill`, but does not require an initialized output range.

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

### **uninitialized\_fill\_n()**

```
template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
```

```

_FowardIterator std::uninitialized_fill_n (
 _FowardIterator __first,
 _Size __n,
 const _Tp & __x) [inline], [constexpr]

```

Copies the value x into the range [first,first+n).

#### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__first</code> | A forward iterator.           |
| <code>__n</code>     | The number of copies to make. |
| <code>__x</code>     | The source value.             |

#### Returns

`__first + __n.`

Like `std::fill_n`, but does not require an initialized output range.

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

#### uninitialized\_move()

```

template<typename _InputIterator, typename _ForwardIterator>
_FowardIterator std::uninitialized_move (
 _InputIterator __first,
 _InputIterator __last,
 _ForwardIterator __result) [inline], [constexpr]

```

Move-construct from the range [first,last) into result.

#### Parameters

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

#### Returns

`__result + (__first - __last)`

#### Since

C++17

References [uninitialized\\_copy\(\)](#).

#### uninitialized\_move\_n()

```

template<typename _InputIterator, typename _Size, typename _ForwardIterator>
pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (
 _InputIterator __first,
 _Size __count,
 _ForwardIterator __result) [inline], [constexpr]

```

Move-construct from the range [first,first+count) into result.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| <code>__first</code>  | An input iterator.                   |
| <code>__count</code>  | The number of objects to initialize. |
| <code>__result</code> | An output iterator.                  |

**Returns**

`__result + __count`

**Since**

C++17

**uninitialized\_value\_construct()**

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

Value-initializes objects in the range [first,last).

**Parameters**

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

**Since**

C++17

**uninitialized\_value\_construct\_n()**

```
template<typename _ForwardIterator, typename _Size>
_FowardIterator std::uninitialized_value_construct_n (
 _ForwardIterator __first,
 _Size __count) [inline], [constexpr]
```

Value-initializes objects in the range [first,first+count).

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__first</code> | A forward iterator.                 |
| <code>__count</code> | The number of objects to construct. |

**Returns**

`__result + __count`

Since

C++17

### 3.16.6.3 Allocators

Collaboration diagram for Allocators:



#### Classes

- struct `__gnu_cxx::__alloc_traits<_Alloc, typename >`
- class `__gnu_cxx::__mt_alloc<_Tp, _Poolp >`
- class `std::__new_allocator<_Tp >`
- class `__gnu_cxx::__pool_alloc<_Tp >`
- class `__gnu_cxx::__ExtPtr_allocator<_Tp >`
- class `std::allocator<_Tp >`
- class `std::allocator<void >`
- struct `std::allocator_traits<_Alloc >`
- struct `std::allocator_traits<allocator<_Tp > >`
- struct `std::allocator_traits<allocator<void > >`
- class `__gnu_cxx::bitmap_allocator<_Tp >`
- class `__gnu_cxx::debug_allocator<_Alloc >`
- class `__gnu_cxx::malloc_allocator<_Tp >`
- class `__gnu_cxx::new_allocator<_Tp >`
- class `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs >`
- class `__gnu_cxx::throw_allocator_base<_Tp, _Cond >`
- struct `std::uses_allocator<typename, typename >`

#### Typedefs

- template<typename \_Tp>  
using `std::__allocator_base`

#### Functions

- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
constexpr `_Tp std::make_obj_using_allocator` (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- template<typename \_T1, typename \_T2>  
constexpr bool `operator==` (const allocator<\_T1 > &, const allocator<\_T2 > &) noexcept
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool `operator==` (const scoped\_allocator\_adaptor<\_OutA1, \_InA... > &\_\_a, const scoped\_allocator\_adaptor<\_OutA2, \_InA... > &\_\_b) noexcept

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp * std::uninitialized_construct_using_allocator (_Tp *__p, const _Alloc &__a, _Args &&... __args)`
- `template<_Std_pair _Tp, typename _Alloc>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, _Up &&, _Vp &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`requires (! _Std_pair<_Tp>)`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, _Args &&... __args) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, piecewise\_construct\_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept`

### 3.16.6.3.1 Detailed Description

Classes encapsulating memory operations.

### 3.16.6.3.2 Typedef Documentation

#### `__allocator_base`

```
template<typename _Tp>
using std::__allocator_base
```

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `std::__new_allocator`.

### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

### 3.16.6.3.3 Function Documentation

#### `operator==()`

```
template<typename _T1, typename _T2>
bool operator== (
 const allocator< _T1 > & ,
 const allocator< _T2 > &) [related]
```

Equality comparison for `std::allocator` objects

#### Returns

true, for all `std::allocator` objects.

### 3.16.6.4 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



#### Classes

- class `std::auto_ptr<_Tp>`
- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<shared_ptr<_Tp>>`
- struct `std::hash<unique_ptr<_Tp, _Dp>>`
- struct `std::owner_less<_Tp>`
- struct `std::owner_less<shared_ptr<_Tp>>`
- struct `std::owner_less<void>`
- struct `std::owner_less<weak_ptr<_Tp>>`
- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`
- class `std::shared_ptr<_Tp>`
- class `std::unique_ptr<_Tp, _Dp>`
- class `std::unique_ptr<_Tp[], _Dp>`
- class `std::weak_ptr<_Tp>`

#### Functions

- template<typename \_Del, typename \_Tp, \_Lock\_policy \_Lp>  
\_Del \* **std::get\_deleter** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_p) noexcept
- template<typename \_Del, typename \_Tp>  
\_Del \* **get\_deleter** (const `shared_ptr<_Tp>` &\_\_p) noexcept
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
constexpr bool **std::operator<** (const `unique_ptr<_Tp, _Dp>` &\_\_x, const `unique_ptr<_Up, _Ep>` &\_\_y)
- template<typename \_Tp, typename \_Dp>  
constexpr bool **std::operator<** (const `unique_ptr<_Tp, _Dp>` &\_\_x, nullptr\_t)
- template<typename \_Tp, typename \_Dp>  
constexpr bool **std::operator<** (nullptr\_t, const `unique_ptr<_Tp, _Dp>` &\_\_x)
- template<typename \_Ch, typename \_Tr, typename \_Tp, \_Lock\_policy \_Lp>  
`std::basic_ostream<_Ch, _Tr>` & **operator<<** (`std::basic_ostream<_Ch, _Tr>` &\_\_os, const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_p)
- template<typename \_CharT, typename \_Traits, typename \_Tp, typename \_Dp>  
`basic_ostream<_CharT, _Traits>` & **operator<<** (`basic_ostream<_CharT, _Traits>` &\_\_os, const `unique_ptr<_Tp, _Dp>` &\_\_p)
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
constexpr bool **std::operator<=** (const `unique_ptr<_Tp, _Dp>` &\_\_x, const `unique_ptr<_Up, _Ep>` &\_\_y)

- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > std::operator<=> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > std::operator<=> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Del>`  
`std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >`
- `template<typename _Tp>`  
`std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >`
- `template<typename _Tp, typename _Dp>`  
`enable_if<!__is_swappable< _Dp >::value >::type std::swap (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete`
- `template<typename _Tp, typename _Dp>`  
`constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp>`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp * std::to_address (_Tp *__ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr auto std::to_address (const _Ptr &__ptr) noexcept`
- `template<typename _Tp>`  
`std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)`
- `template<typename _Tp>`  
`bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)`



- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp>`  
`void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp>`  
`void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, typename _Up>`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`

- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)`

#### 3.16.6.4.1 Detailed Description

Smart pointers, etc.

#### 3.16.6.4.2 Function Documentation

##### **allocate\_shared()**

```
template<typename _Tp, typename _Alloc, typename... _Args>
shared_ptr< _NonArray< _Tp > > allocate_shared (
 const _Alloc & __a,
 _Args &&... __args) [related]
```

Create an object that is owned by a `shared_ptr`.

##### **Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__a</code>    | An allocator.                                            |
| <code>__args</code> | Arguments for the <code>_Tp</code> object's constructor. |

**Returns**

A `shared_ptr` that owns the newly created object.

**Exceptions**

|           |                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------|
| <i>An</i> | exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> . |
|-----------|---------------------------------------------------------------------------------------------------|

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

**atomic\_compare\_exchange\_strong() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > * __v,
 __shared_ptr< _Tp, _Lp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| <code>__v</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| <code>__w</code> | A non-null pointer to a <code>shared_ptr</code> object. |

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_strong() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > * __v,
 shared_ptr< _Tp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| <code>__v</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| <code>__w</code> | A non-null pointer to a <code>shared_ptr</code> object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

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

**atomic\_compare\_exchange\_strong\_explicit()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong_explicit (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > * __v,
 __shared_ptr< _Tp, _Lp > __w,
 memory_order ,
 memory_order)
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

|                                                                                          |                                            |
|------------------------------------------------------------------------------------------|--------------------------------------------|
| <br>__p | A non-null pointer to a shared_ptr object. |
| <br>__v | A non-null pointer to a shared_ptr object. |
| <br>__w | A non-null pointer to a shared_ptr object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

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

**atomic\_compare\_exchange\_strong\_explicit()** [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong_explicit (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > * __v,
 shared_ptr< _Tp > __w,
 memory_order ,
 memory_order)
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

|                                                                                            |                                            |
|--------------------------------------------------------------------------------------------|--------------------------------------------|
| <br>__p | A non-null pointer to a shared_ptr object. |
| <br>__v | A non-null pointer to a shared_ptr object. |
| <br>__w | A non-null pointer to a shared_ptr object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > * __v,
 __shared_ptr< _Tp, _Lp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| $\leftarrow$<br>__p | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br>__v | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br>__w | A non-null pointer to a <code>shared_ptr</code> object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_compare\_exchange\_weak() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > * __v,
 shared_ptr< _Tp > __w) [inline]
```

Atomic compare-and-swap for `shared_ptr` objects.

**Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| $\leftarrow$<br>__p | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br>__v | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br>__w | A non-null pointer to a <code>shared_ptr</code> object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

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

**atomic\_compare\_exchange\_weak\_explicit() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak_explicit (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > * __v,
 __shared_ptr< _Tp, _Lp > __w,
 memory_order __success,
 memory_order __failure) [inline]
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

|                                                                                          |                                            |
|------------------------------------------------------------------------------------------|--------------------------------------------|
| <br>__p | A non-null pointer to a shared_ptr object. |
| <br>__v | A non-null pointer to a shared_ptr object. |
| <br>__w | A non-null pointer to a shared_ptr object. |

**Returns**

True if \*\_\_p was equivalent to \*\_\_v, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.

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

**atomic\_compare\_exchange\_weak\_explicit() [2/2]**

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak_explicit (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > * __v,
 shared_ptr< _Tp > __w,
 memory_order __success,
 memory_order __failure) [inline]
```

Atomic compare-and-swap for shared\_ptr objects.

**Parameters**

|                                                                                            |                                            |
|--------------------------------------------------------------------------------------------|--------------------------------------------|
| <br>__p | A non-null pointer to a shared_ptr object. |
| <br>__v | A non-null pointer to a shared_ptr object. |
| <br>__w | A non-null pointer to a shared_ptr object. |

**Returns**

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.  
References [move\(\)](#).

**atomic\_exchange()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > __r) [inline]
```

Atomic exchange for `shared_ptr` objects.

**Parameters**

|                                 |                                                         |
|---------------------------------|---------------------------------------------------------|
| $\leftarrow$<br><code>_p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br><code>_r</code> | New value to store in <code>*__p</code> .               |

**Returns**

The original value of `*__p`

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

**atomic\_exchange()** [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > __r) [inline]
```

Atomic exchange for `shared_ptr` objects.

**Parameters**

|                                 |                                                         |
|---------------------------------|---------------------------------------------------------|
| $\leftarrow$<br><code>_p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br><code>_r</code> | New value to store in <code>*__p</code> .               |

**Returns**

The original value of `*__p`

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

**atomic\_exchange\_explicit()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > __r,
 memory_order) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

|                                                                                                       |                                            |
|-------------------------------------------------------------------------------------------------------|--------------------------------------------|
| <br><code>__p</code> | A non-null pointer to a shared_ptr object. |
| <br><code>__r</code> | New value to store in *__p.                |

**Returns**

The original value of \*\_\_p

**atomic\_exchange\_explicit()** [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange_explicit (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > __r,
 memory_order) [inline]
```

Atomic exchange for shared\_ptr objects.

**Parameters**

|                                                                                                         |                                            |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------|
| <br><code>__p</code> | A non-null pointer to a shared_ptr object. |
| <br><code>__r</code> | New value to store in *__p.                |

**Returns**

The original value of \*\_\_p

**atomic\_is\_lock\_free()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_is_lock_free (
 const __shared_ptr< _Tp, _Lp > *) [inline]
```



Report whether `shared_ptr` atomic operations are lock-free.

#### Parameters

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|------------------|---------------------------------------------------------|

#### Returns

True if atomic access to `*__p` is lock-free, false otherwise.

#### `atomic_is_lock_free()` [2/2]

```
template<typename _Tp>
bool std::atomic_is_lock_free (
 const shared_ptr< _Tp > * __p) [inline]
```

Report whether `shared_ptr` atomic operations are lock-free.

#### Parameters

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|------------------|---------------------------------------------------------|

#### Returns

True if atomic access to `*__p` is lock-free, false otherwise.

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

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load (
 const __shared_ptr< _Tp, _Lp > * __p) [inline]
```

Atomic load for `shared_ptr` objects.

#### Parameters

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|------------------|---------------------------------------------------------|

#### Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

#### `atomic_load()` [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load (
 const shared_ptr< _Tp > * __p) [inline]
```

Atomic load for `shared_ptr` objects.

**Parameters**

|                                  |                                                         |
|----------------------------------|---------------------------------------------------------|
| $\leftarrow$<br><code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|----------------------------------|---------------------------------------------------------|

**Returns**

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_load\_explicit() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (
 const __shared_ptr< _Tp, _Lp > * __p,
 memory_order) [inline]
```

Atomic load for `shared_ptr` objects.

**Parameters**

|                                  |                                                         |
|----------------------------------|---------------------------------------------------------|
| $\leftarrow$<br><code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|----------------------------------|---------------------------------------------------------|

**Returns**

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_load\_explicit() [2/2]**

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load_explicit (
 const shared_ptr< _Tp > * __p,
 memory_order) [inline]
```

Atomic load for `shared_ptr` objects.

**Parameters**

|                                  |                                                         |
|----------------------------------|---------------------------------------------------------|
| $\leftarrow$<br><code>__p</code> | A non-null pointer to a <code>shared_ptr</code> object. |
|----------------------------------|---------------------------------------------------------|

**Returns**

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

**atomic\_store() [1/2]**

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store (
```

```
__shared_ptr< _Tp, _Lp > * __p,
__shared_ptr< _Tp, _Lp > __r) [inline]
```

Atomic store for shared\_ptr objects.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| $\leftarrow$<br>_p | A non-null pointer to a shared_ptr object. |
| $\leftarrow$<br>_r | The value to store.                        |

The memory order shall not be memory\_order\_acquire or memory\_order\_acq\_rel.

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

#### atomic\_store() [2/2]

```
template<typename _Tp>
void std::atomic_store (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > __r) [inline]
```

Atomic store for shared\_ptr objects.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| $\leftarrow$<br>_p | A non-null pointer to a shared_ptr object. |
| $\leftarrow$<br>_r | The value to store.                        |

The memory order shall not be memory\_order\_acquire or memory\_order\_acq\_rel.

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

#### atomic\_store\_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store_explicit (
 __shared_ptr< _Tp, _Lp > * __p,
 __shared_ptr< _Tp, _Lp > __r,
 memory_order) [inline]
```

Atomic store for shared\_ptr objects.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| $\leftarrow$<br>_p | A non-null pointer to a shared_ptr object. |
|--------------------|--------------------------------------------|

|              |                     |
|--------------|---------------------|
| $\leftarrow$ | The value to store. |
| $r$          |                     |

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

### **atomic\_store\_explicit()** [2/2]

```
template<typename _Tp>
void std::atomic_store_explicit (
 shared_ptr< _Tp > * __p,
 shared_ptr< _Tp > __r,
 memory_order) [inline]
```

Atomic store for `shared_ptr` objects.

#### **Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| $\leftarrow$<br>$p$ | A non-null pointer to a <code>shared_ptr</code> object. |
| $\leftarrow$<br>$r$ | The value to store.                                     |

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

### **const\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
 const shared_ptr< _Up > & __r) [related]
```

Convert type of `shared_ptr`, via `const_cast`.

### **const\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
 shared_ptr< _Up > && __r) [related]
```

Convert type of `shared_ptr` rvalue, via `const_cast`.

Since

C++20

### **dynamic\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
 const shared_ptr< _Up > & __r) [related]
```

Convert type of `shared_ptr`, via `dynamic_cast`.

**dynamic\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
 shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via dynamic\_cast.

Since

C++20

**get\_deleter()**

```
template<typename _Del, typename _Tp>
_Del * get_deleter (
 const shared_ptr< _Tp > & __p) [related]
```

## 20.7.2.2.10 shared\_ptr get\_deleter

If \_\_p has a deleter of type \_Del, return a pointer to it.

**make\_shared()**

```
template<typename _Tp, typename... _Args>
shared_ptr< _NonArray< _Tp > > make_shared (
 _Args &&... __args) [related]
```

Create an object that is owned by a shared\_ptr.

**Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__args</code> | Arguments for the <code>_Tp</code> object's constructor. |
|---------------------|----------------------------------------------------------|

**Returns**

A shared\_ptr that owns the newly created object.

**Exceptions**

|                                  |                                                                |
|----------------------------------|----------------------------------------------------------------|
| <code>std::bad_alloc</code> , or | an exception thrown from the constructor of <code>_Tp</code> . |
|----------------------------------|----------------------------------------------------------------|

**operator<()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator< (
 const unique_ptr< _Tp, _Dp > & __x,
 const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique\_ptr objects, compares the owned pointers.

References `std::unique_ptr< _Tp, _Dp >::get()`.

**operator<()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
 const unique_ptr< _Tp, _Dp > & __x,
 nullptr_t) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

References [std::unique\\_ptr<\\_Tp, \\_Dp>::get\(\)](#).

### **operator<()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
 nullptr_t ,
 const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

References [std::unique\\_ptr<\\_Tp, \\_Dp>::get\(\)](#).

### **operator<<()** [1/2]

```
template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
std::basic_ostream< _Ch, _Tr > & operator<< (
 std::basic_ostream< _Ch, _Tr > & __os,
 const __shared_ptr< _Tp, _Lp > & __p) [related]
```

Write the stored pointer to an ostream.

### **operator<<()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
basic_ostream< _CharT, _Traits > & operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const unique_ptr< _Tp, _Dp > & __p) [related]
```

Stream output operator for unique\_ptr.

Since

C++20

### **operator<=()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator<= (
 const unique_ptr< _Tp, _Dp > & __x,
 const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique\_ptr objects, compares the owned pointers.

### **operator<=()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
 const unique_ptr< _Tp, _Dp > & __x,
 nullptr_t) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

### **operator<=()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
 nullptr_t ,
 const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

**operator==()** [1/4]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator== (
 const unique_ptr< _Tp, _Dp > & __x,
 const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Equality operator for [unique\\_ptr](#) objects, compares the owned pointers.

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator==()** [2/4]

```
template<typename _Tp, typename _Dp>
bool std::operator== (
 const unique_ptr< _Tp, _Dp > & __x,
 nullptr_t) [inline], [nodiscard], [constexpr], [noexcept]
```

[unique\\_ptr](#) comparison with [nullptr](#)

**operator==()** [3/4]

```
template<typename _Tp, typename _Up>
bool operator== (
 const shared_ptr< _Tp > & __a,
 const shared_ptr< _Up > & __b) [related]
```

Equality operator for [shared\\_ptr](#) objects, compares the stored pointers.

**operator==()** [4/4]

```
template<typename _Tp>
bool operator== (
 const shared_ptr< _Tp > & __a,
 nullptr_t) [related]
```

[shared\\_ptr](#) comparison with [nullptr](#)

**operator>()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator> (
 const unique_ptr< _Tp, _Dp > & __x,
 const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for [unique\\_ptr](#) objects, compares the owned pointers.

**operator>()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
 const unique_ptr< _Tp, _Dp > & __x,
 nullptr_t) [inline], [nodiscard], [constexpr]
```

[unique\\_ptr](#) comparison with [nullptr](#)

References [std::unique\\_ptr< \\_Tp, \\_Dp >::get\(\)](#).

**operator>()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
 nullptr_t ,
 const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

References [std::unique\\_ptr<\\_Tp, \\_Dp>::get\(\)](#).

### **operator>=()** [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator>= (
 const unique_ptr< _Tp, _Dp > & __x,
 const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique\_ptr objects, compares the owned pointers.

### **operator>=()** [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
 const unique_ptr< _Tp, _Dp > & __x,
 nullptr_t) [inline], [nodiscard], [constexpr]
```

unique\_ptr comparison with nullptr

### **operator>=()** [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
 nullptr_t ,
 const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard]
```

unique\_ptr comparison with nullptr

### **reinterpret\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
 const shared_ptr< _Up > & __r) [related]
```

Convert type of shared\_ptr, via reinterpret\_cast.

Since

C++17

### **reinterpret\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
 shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via reinterpret\_cast.

Since

C++20

### **static\_pointer\_cast()** [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
 const shared_ptr< _Up > & __r) [related]
```

Convert type of shared\_ptr, via static\_cast.



**static\_pointer\_cast()** [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
 shared_ptr< _Up > && __r) [related]
```

Convert type of shared\_ptr rvalue, via static\_cast.

Since

C++20

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

```
template<typename _Tp>
void swap (
 shared_ptr< _Tp > & __a,
 shared_ptr< _Tp > & __b) [related]
```

Swap overload for shared\_ptr.

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

```
template<typename _Tp, typename _Dp>
enable_if< __is_swappable< _Dp >::value >::type swap (
 unique_ptr< _Tp, _Dp > & __x,
 unique_ptr< _Tp, _Dp > & __y) [related]
```

Swap overload for unique\_ptr.

**swap()** [3/3]

```
template<typename _Tp>
void swap (
 weak_ptr< _Tp > & __a,
 weak_ptr< _Tp > & __b) [related]
```

Swap overload for weak\_ptr.

**to\_address()** [1/2]

```
template<typename _Tp>
_Tp * std::to_address (
 _Tp * __ptr) [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <code>__ptr</code> | A pointer to an object |
|--------------------|------------------------|

**Returns**

`__ptr`

Referenced by `std::basic_string< _CharT >::assign()`, `find()`, `to_address()`, `uninitialized_copy()`, `uninitialized_fill()`, and `uninitialized_fill_n()`.

**to\_address()** [2/2]

```
template<typename _Ptr>
auto std::to_address (
 const _Ptr & __ptr) [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <code>__ptr</code> | A pointer to an object |
|--------------------|------------------------|

**Returns**

`pointer_traits<_Ptr>::to_address(__ptr)` if that expression is well-formed, otherwise `to_address(__ptr.operator->())`.

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

**3.16.6.5 Polymorphic memory resources**

Collaboration diagram for Polymorphic memory resources:

**Files**

- file [memory\\_resource](#)

**Classes**

- struct [std::allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >](#)
- class [std::pmr::memory\\_resource](#)
- class [std::pmr::monotonic\\_buffer\\_resource](#)
- class [std::pmr::polymorphic\\_allocator< \\_Tp >](#)
- struct [std::pmr::pool\\_options](#)
- class [std::pmr::unsynchronized\\_pool\\_resource](#)

**Functions**

- [memory\\_resource \\* std::pmr::new\\_delete\\_resource \(\)](#) noexcept

**3.16.6.5.1 Detailed Description**

Since

C++17

Memory resources are classes that implement the `std::pmr::memory_resource` interface for allocating and deallocating memory. Unlike traditional C++ allocators, memory resources are not value types and are used via pointers to the abstract base class. They are only responsible for allocating and deallocating, not for construction and destruction of objects. As a result, memory resources just allocate raw memory as type `void*` and are not templates that allocate/deallocate and construct/destroy a specific type.

The class template `std::pmr::polymorphic_allocator` is an allocator that uses a memory resource for its allocations.

### 3.16.6.5.2 Function Documentation

#### `new_delete_resource()`

```
memory_resource * std::pmr::new_delete_resource () [nodiscard], [noexcept]
```

A `pmr::memory_resource` that uses `new` to allocate memory.

Since

C++17

### 3.16.7 Metaprogramming

Collaboration diagram for Metaprogramming:



#### Topics

- [Variable Templates](#)

#### Classes

- `struct std::tr2::__reflection_typelist< _Elements >`
- `struct std::tr2::__reflection_typelist< _First, _Rest... >`
- `struct std::tr2::__reflection_typelist<>`
- `struct std::add_const< _Tp >`
- `struct std::add_cv< _Tp >`
- `struct std::add_lvalue_reference< _Tp >`
- `struct std::add_pointer< _Tp >`
- `struct std::add_rvalue_reference< _Tp >`
- `struct std::add_volatile< _Tp >`
- `struct std::aligned_storage< _Len, _Align >`
- `struct std::aligned_union< _Len, _Types >`
- `struct std::alignment_of< _Tp >`
- `struct std::tr2::bases< _Tp >`
- `struct std::common_type< _Tp >`

- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- struct `std::tr2::direct_bases< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _Uint >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< _Tp >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< _Tp >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`
- struct `std::is_default_constructible< _Tp >`
- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_layout_compatible< _Tp, _Up >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_pointer_interconvertible_base_of< _Base, _Derived >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_signed< _Tp >`

- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_copyable< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_unsigned< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< _Tp >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`
- struct `std::remove_extent< _Tp >`
- struct `std::remove_pointer< _Tp >`
- struct `std::remove_reference< _Tp >`
- struct `std::remove_volatile< _Tp >`
- struct `std::result_of< _Signature >`
- struct `std::underlying_type< _Tp >`

### Typedefs

- template<bool \_Cond, typename \_If, typename \_Else>  
using `std::__conditional_t`
- template<bool \_Cond, typename \_Tp = void>  
using `std::__enable_if_t`
- template<typename \_ToElementType, typename \_FromElementType>  
using `std::__is_array_convertible`
- using `std::__Build_index_tuple< _Num >::__type`
- template<typename, size\_t... \_Indices>  
using `std::__Build_index_tuple< _Num >::__IdxTuple`
- template<typename \_Tp>  
using `std::add_lvalue_reference_t`
- template<typename \_Tp>  
using `std::add_pointer_t`
- template<typename \_Tp>  
using `std::add_rvalue_reference_t`
- template<size\_t \_Len, size\_t \_Align = \_\_aligned\_storage\_default\_alignment(\_Len)>  
using `std::aligned_storage_t`
- template<size\_t \_Len, typename... \_Types>  
using `std::aligned_union_t`
- template<typename... \_Tp>  
using `std::common_type_t`

- `template<bool _Cond, typename _Iftrue, typename _Iffalse>`  
  using `std::conditional_t`
- `template<typename _Tp>`  
  using `std::decay_t`
- `template<bool _Cond, typename _Tp = void>`  
  using `std::enable_if_t`
- using `std::false_type`
- `template<typename _Tp>`  
  using `std::make_signed_t`
- `template<typename _Tp>`  
  using `std::make_unsigned_t`
- `template<typename _Tp>`  
  using `std::remove_all_extents_t`
- `template<typename _Tp>`  
  using `std::remove_extent_t`
- `template<typename _Tp>`  
  using `std::remove_pointer_t`
- `template<typename _Tp>`  
  using `std::remove_reference_t`
- `template<typename _Tp>`  
  using `std::result_of_t`
- using `std::true_type`
- `template<typename _Tp, typename>`  
  using `std::__conditional< bool >::type`
- `template<typename, typename _Up>`  
  using `std::__conditional< false >::type`
- using `std::__is_implicitly_default_constructible_impl< _Tp >::type`
- using `std::add_const< _Tp >::type`
- using `std::add_cv< _Tp >::type`
- using `std::add_lvalue_reference< _Tp >::type`
- using `std::add_pointer< _Tp >::type`
- using `std::add_rvalue_reference< _Tp >::type`
- using `std::add_volatile< _Tp >::type`
- using `std::aligned_union< _Len, _Types >::type`
- using `std::common_reference< _Tp0 >::type`
- using `std::conditional< _Cond, _Iftrue, _Iffalse >::type`
- using `std::conditional< false, _Iftrue, _Iffalse >::type`
- using `std::decay< _Tp >::type`
- using `std::enable_if< true, _Tp >::type`
- using `std::integral_constant< _Tp, __v >::type`
- using `std::make_signed< _Tp >::type`
- using `std::make_unsigned< _Tp >::type`
- using `std::remove_all_extents< _Tp >::type`
- using `std::remove_const< _Tp >::type`
- using `std::remove_const< _Tp const >::type`
- using `std::remove_cv< _Tp >::type`
- using `std::remove_extent< _Tp >::type`
- using `std::remove_pointer< _Tp >::type`
- using `std::remove_reference< _Tp >::type`
- using `std::remove_volatile< _Tp >::type`
- using `std::remove_volatile< _Tp volatile >::type`
- `template<typename _Tp>`  
  using `std::underlying_type_t`
- using `std::integral_constant< _Tp, __v >::value_type`

## Functions

- `template<typename _Tp>`  
`static void std::do_is_implicitly_default_constructible_impl::__helper (const _Tp &)`
- `static false_type std::do_is_implicitly_default_constructible_impl::__test (...)`
- `template<typename _Tp>`  
`static true_type std::do_is_implicitly_default_constructible_impl::__test (const _Tp &, decltype(__helper< const _Tp & >({})) *=0)`
- `constexpr std::integral_constant< _Tp, __v >::operator value_type () const noexcept`
- `template<typename _Tp>`  
`constexpr Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr __enable_if_t< __is_swappable< _Tp >::value > std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm])↔`  
`) noexcept(__is_nothrow_swappable< _Tp >::value)`

## Variables

- `unsigned char std::aligned_storage< _Len, _Align >::type::__data [_Len]`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::is_nothrow_new_constructible`
- `static const size_t std::__strictest_alignment< _Types >::S_alignment`
- `static const size_t std::__strictest_alignment< _Tp, _Types... >::S_alignment`
- `static const size_t std::__strictest_alignment< _Types >::S_size`
- `static const size_t std::__strictest_alignment< _Tp, _Types... >::S_size`
- `template<size_t _Len, typename... _Types>`  
`const size_t std::aligned_union< _Len, _Types... >::alignment_value`
- `static const size_t std::aligned_union< _Len, _Types >::alignment_value`
- `static constexpr _Tp std::integral_constant< _Tp, __v >::value`
- `template<typename... _Tp>`  
`using std::common_reference_t`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::is_layout_compatible_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool std::is_pointer_interconvertible_base_of_v`
- `template<typename _S1, typename _S2, typename _M1, typename _M2>`  
`constexpr bool std::is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept`
- `template<typename _Tp, typename _Mem>`  
`constexpr bool std::is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept`

### 3.16.7.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

Since

C++11

### 3.16.7.2 Typedef Documentation

#### **add\_lvalue\_reference\_t**

```
template<typename _Tp>
using std::add_lvalue_reference_t
Alias template for add_lvalue_reference.
```

#### **add\_pointer\_t**

```
template<typename _Tp>
using std::add_pointer_t
Alias template for add_pointer.
```

#### **add\_rvalue\_reference\_t**

```
template<typename _Tp>
using std::add_rvalue_reference_t
Alias template for add_rvalue_reference.
```

#### **aligned\_storage\_t**

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
using std::aligned_storage_t
Alias template for aligned_storage.
```

#### **common\_reference\_t**

```
template<typename... _Tp>
using std::common_reference_t
Since
```

C++20

#### **common\_type\_t**

```
template<typename... _Tp>
using std::common_type_t
Alias template for common_type.
```

#### **conditional\_t**

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
using std::conditional_t
Alias template for conditional.
```

#### **decay\_t**

```
template<typename _Tp>
using std::decay_t
Alias template for decay.
```

#### **enable\_if\_t**

```
template<bool _Cond, typename _Tp = void>
using std::enable_if_t
Alias template for enable_if.
```



**false\_type**

```
using std::false_type
```

The type used as a compile-time boolean with false value.

**make\_signed\_t**

```
template<typename _Tp>
```

```
using std::make_signed_t
```

Alias template for make\_signed.

**make\_unsigned\_t**

```
template<typename _Tp>
```

```
using std::make_unsigned_t
```

Alias template for make\_unsigned.

**remove\_all\_extents\_t**

```
template<typename _Tp>
```

```
using std::remove_all_extents_t
```

Alias template for remove\_all\_extents.

**remove\_extent\_t**

```
template<typename _Tp>
```

```
using std::remove_extent_t
```

Alias template for remove\_extent.

**remove\_pointer\_t**

```
template<typename _Tp>
```

```
using std::remove_pointer_t
```

Alias template for remove\_pointer.

**remove\_reference\_t**

```
template<typename _Tp>
```

```
using std::remove_reference_t
```

Alias template for remove\_reference.

**result\_of\_t**

```
template<typename _Tp>
```

```
using std::result_of_t
```

Alias template for result\_of.

**true\_type**

```
using std::true_type
```

The type used as a compile-time boolean with true value.

**type**

```
template<size_t _Len, typename... _Types>
using std::aligned_union< _Len, _Types >::type
The storage.
```

**underlying\_type\_t**

```
template<typename _Tp>
using std::underlying_type_t
Alias template for underlying_type.
```

**3.16.7.3 Function Documentation****is\_corresponding\_member()**

```
template<typename _S1, typename _S2, typename _M1, typename _M2>
bool std::is_corresponding_member (
 _M1 _S1::* __m1,
 _M2 _S2::* __m2) [constexpr], [noexcept]
```

**Since**

C++20

**is\_pointer\_interconvertible\_with\_class()**

```
template<typename _Tp, typename _Mem>
bool std::is_pointer_interconvertible_with_class (
 _Mem _Tp::* __mp) [constexpr], [noexcept]
```

True if `__mp` points to the first member of a standard-layout type.**Returns**true if `s.*__mp` is pointer-interconvertible with `s`**Since**

C++20

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

```
template<typename _Tp>
_Require< __not< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
_Tp > > std::swap (
 _Tp & __a,
 _Tp & __b) [inline], [constexpr], [noexcept]
```

Swaps two values.

**Parameters**

|                             |                                  |
|-----------------------------|----------------------------------|
| $\longleftrightarrow$<br>_a | A thing of arbitrary type.       |
| $\longleftrightarrow$<br>_b | Another thing of arbitrary type. |

**swap()** [2/2]

```
template<typename _Tp, size_t _Nm>
__enable_if_t< __is_swappable< _Tp >::value > std::swap (
 _Tp(&) __a[_Nm],
 _Tp(&) __b[_Nm]) [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

**3.16.7.4 Variable Documentation****alignment\_value**

```
template<size_t _Len, typename... _Types>
const size_t std::aligned_union< _Len, _Types >::alignment_value [static]
```

The value of the strictest alignment of \_Types.

**is\_layout\_compatible\_v**

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v [constexpr]
```

Since

C++20

**is\_pointer\_interconvertible\_base\_of\_v**

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v [constexpr]
```

Since

C++20

**3.16.7.5 Variable Templates**

Collaboration diagram for Variable Templates:

**Variables**

- `template<typename _Tp, typename _Up>`  
`constexpr strong_ordering std::type_order_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::is_layout_compatible_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool std::is_pointer_interconvertible_base_of_v`

### 3.16.7.5.1 Detailed Description

Since

C++17

Each trait in `<type_traits>` that has a `value` static data member is also available as a variable template, using the same name with a `_v` suffix. For example, `std::is_integral_v<T>` is a `bool` constant with the same value as `std::is_integral<T>::value`.

In some cases, using the variable template instead of the `value` member avoids instantiating the class template for the trait. This gives the compiler less work to do and can make your code compile a little faster.

### 3.16.7.5.2 Variable Documentation

#### **is\_layout\_compatible\_v**

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v [constexpr]
```

Since

C++20

#### **is\_pointer\_interconvertible\_base\_of\_v**

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v [constexpr]
```

Since

C++20

#### **type\_order\_v**

```
template<typename _Tp, typename _Up>
strong_ordering std::type_order_v [inline], [constexpr]
```

Since

C++26

## 3.16.8 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



### Files

- file [ratio](#)

## Classes

- struct `std::ratio<_Num, _Den >`
- struct `std::ratio_equal<_R1, _R2 >`
- struct `std::ratio_greater<_R1, _R2 >`
- struct `std::ratio_greater_equal<_R1, _R2 >`
- struct `std::ratio_less<_R1, _R2 >`
- struct `std::ratio_less_equal<_R1, _R2 >`
- struct `std::ratio_not_equal<_R1, _R2 >`

## Typedefs

- using `std::atto`
- using `std::centi`
- using `std::deca`
- using `std::deci`
- using `std::exa`
- using `std::femto`
- using `std::giga`
- using `std::hecto`
- using `std::kilo`
- using `std::mega`
- using `std::micro`
- using `std::milli`
- using `std::nano`
- using `std::peta`
- using `std::pico`
- template<typename \_R1, typename \_R2>  
using `std::ratio_add`
- template<typename \_R1, typename \_R2>  
using `std::ratio_divide`
- template<typename \_R1, typename \_R2>  
using `std::ratio_multiply`
- template<typename \_R1, typename \_R2>  
using `std::ratio_subtract`
- using `std::tera`

## Variables

- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_equal_v`
- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_greater_equal_v`
- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_greater_v`
- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_less_equal_v`
- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_less_v`
- template<typename \_R1, typename \_R2>  
constexpr bool `std::ratio_not_equal_v`

### 3.16.8.1 Detailed Description

Compile time representation of finite rational numbers.

### 3.16.8.2 Typedef Documentation

#### ratio\_add

```
template<typename _R1, typename _R2>
using std::ratio_add
ratio_add
```

#### ratio\_divide

```
template<typename _R1, typename _R2>
using std::ratio_divide
ratio_divide
```

#### ratio\_multiply

```
template<typename _R1, typename _R2>
using std::ratio_multiply
ratio_multiply
```

#### ratio\_subtract

```
template<typename _R1, typename _R2>
using std::ratio_subtract
ratio_subtract
```

### 3.16.9 Time

Collaboration diagram for Time:



#### Files

- file [chrono](#)

#### Namespaces

- namespace [std::chrono](#)
- namespace [std::literals::chrono\\_literals](#)

## Classes

- struct `std::common_type< chrono::duration< _Rep, _Period > >`
- struct `std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >`
- struct `std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >`
- struct `std::common_type< chrono::time_point< _Clock, _Duration > >`
- struct `std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >`
- struct `std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >`
- class `std::chrono::duration< _Rep, _Period >`
- struct `std::chrono::duration_values< _Rep >`
- class `std::chrono::gps_clock`
- class `std::chrono::hh_mm_ss< _Duration >`
- struct `std::chrono::steady_clock`
- struct `std::chrono::system_clock`
- class `std::chrono::tai_clock`
- class `std::chrono::time_point< _Clock, _Dur >`
- struct `std::chrono::treat_as_floating_point< _Rep >`
- class `std::chrono::tzdb_list`
- class `std::chrono::utc_clock`

## Typedefs

- using `std::chrono::days`
- using `std::chrono::file_clock`
- template<typename \_Duration>  
using `std::chrono::file_time`
- using `std::chrono::gps_seconds`
- template<typename \_Duration>  
using `std::chrono::gps_time`
- using `std::chrono::high_resolution_clock`
- using `std::chrono::hours`
- using `std::chrono::local_days`
- using `std::chrono::local_seconds`
- template<typename \_Duration>  
using `std::chrono::local_time`
- using `std::chrono::microseconds`
- using `std::chrono::milliseconds`
- using `std::chrono::minutes`
- using `std::chrono::months`
- using `std::chrono::nanoseconds`
- using `std::chrono::seconds`
- using `std::chrono::sys_days`
- using `std::chrono::sys_seconds`
- template<typename \_Duration>  
using `std::chrono::sys_time`
- using `std::chrono::tai_seconds`
- template<typename \_Duration>  
using `std::chrono::tai_time`
- using `std::chrono::utc_seconds`
- template<typename \_Duration>  
using `std::chrono::utc_time`
- using `std::chrono::weeks`
- using `std::chrono::years`
- using `std::chrono::zoned_seconds`

## Enumerations

- enum class **choose** { **earliest** , **latest** }

## Functions

- constexpr **std::chrono::year\_month\_day::year\_month\_day** (const year\_month\_day\_last &\_\_ymdl) noexcept
- template<typename \_Rep, typename \_Period>  
constexpr **enable\_if\_t< numeric\_limits< \_Rep >::is\_signed, duration< \_Rep, \_Period > >** **std::chrono::abs** (**duration< \_Rep, \_Period >** \_\_d)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr **\_\_enable\_if\_is\_duration< \_ToDur >** **std::chrono::ceil** (const **duration< \_Rep, \_Period >** &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr **enable\_if\_t< \_\_is\_duration\_v< \_ToDur >, time\_point< \_Clock, \_ToDur > >** **std::chrono::ceil** (const **time\_point< \_Clock, \_Dur >** &\_\_tp)
- template<typename \_DestClock, typename \_SourceClock, typename \_Duration>  
requires \_\_detail::\_\_clock\_convs< \_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_sys< \_DestClock, \_SourceClock, \_\_←  
\_Duration> || \_\_detail::\_\_clock\_convs\_utc< \_DestClock, \_SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_sys\_utc< \_DestClock, \_\_←  
SourceClock, \_Duration> || \_\_detail::\_\_clock\_convs\_utc\_sys< \_DestClock, \_SourceClock, \_Duration>  
auto **std::chrono::clock\_cast** (const **time\_point< \_SourceClock, \_Duration >** &\_\_t)
- const time\_zone \* **std::chrono::current\_zone** ()
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr **\_\_enable\_if\_is\_duration< \_ToDur >** **std::chrono::duration\_cast** (const **duration< \_Rep, \_Period >** &\_\_←  
\_\_d)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr **\_\_enable\_if\_is\_duration< \_ToDur >** **std::chrono::floor** (const **duration< \_Rep, \_Period >** &\_\_d)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr **enable\_if\_t< \_\_is\_duration\_v< \_ToDur >, time\_point< \_Clock, \_ToDur > >** **std::chrono::floor** (const **time\_point< \_Clock, \_Dur >** &\_\_tp)
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is, const  
\_CharT \* \_\_fmt, day & \_\_d, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr, **minutes** \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Rep, typename \_Period, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is, const  
\_CharT \* \_\_fmt, **duration< \_Rep, \_Period >** & \_\_d, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr,  
**minutes** \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is, const  
\_CharT \* \_\_fmt, file\_time< \_Duration > & \_\_tp, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr,  
**minutes** \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is, const  
\_CharT \* \_\_fmt, gps\_time< \_Duration > & \_\_tp, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr,  
**minutes** \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is, const  
\_CharT \* \_\_fmt, local\_time< \_Duration > & \_\_tp, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr,  
**minutes** \* \_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>>  
**basic\_istream< \_CharT, \_Traits >** & **std::chrono::from\_stream** (**basic\_istream< \_CharT, \_Traits >** &\_\_is,  
const \_CharT \* \_\_fmt, month & \_\_m, **basic\_string< \_CharT, \_Traits, \_Alloc >** \* \_\_abbrev=nullptr, **minutes** \* \_\_←  
\_\_offset=nullptr)



- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, month_day & __md, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, sys_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr,`  
`minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is,`  
`const _CharT * __fmt, tai_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr,`  
`minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, utc_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr,`  
`minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, weekday & __wd, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year & __y, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year_month & __ym, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year_month_day & __ymd, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes`  
`* __offset=nullptr)`
- `template<typename _Duration>`  
`static utc_time< common_type_t< _Duration, seconds > > std::chrono::utc_clock::from_sys (const sys_`  
`time< _Duration > & __t)`
- `template<typename _Duration>`  
`leap_second_info std::chrono::get_leap_second_info (const utc_time< _Duration > & __ut)`
- `const tzdb & std::chrono::get_tzdb ()`
- `tzdb_list & std::chrono::get_tzdb_list ()`
- `constexpr bool std::chrono::is_am (const hours & __h) noexcept`
- `constexpr bool std::chrono::is_pm (const hours & __h) noexcept`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const`  
`string * __abbrev=nullptr, const seconds * __offset_sec=nullptr)`
- `const time_zone * std::chrono::locate_zone (string_view __tz_name)`
- `constexpr hours std::chrono::make12 (const hours & __h) noexcept`
- `constexpr hours std::chrono::make24 (const hours & __h, bool __is_pm) noexcept`
- `constexpr bool std::chrono::year_month_day::ok () const noexcept`
- `constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept`
- `template<char... _Digits>`  
`constexpr chrono::hours std::literals::chrono_literals::operator""h ()`
- `constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long dou-`  
`ble __hours)`

- `template<char... _Digits>`  
`constexpr chrono::minutes std::literals::chrono_literals::operator""min ()`
- `constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)`
- `template<char... _Digits>`  
`constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()`
- `constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __← msecs)`
- `template<char... _Digits>`  
`constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()`
- `constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)`
- `template<char... _Digits>`  
`constexpr chrono::seconds std::literals::chrono_literals::operator""s ()`
- `constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)`
- `template<char... _Digits>`  
`constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()`
- `constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __← usecs)`
- `constexpr chrono::year std::literals::chrono_literals::operator""y (unsigned long long __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const day &__d)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`requires requires(const sys_time<_Duration>& __st) { __os << __st; }`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month &__m)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const sys_days &__dp)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`sys_info &__i)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const sys_time< _Duration > &__tp)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const tai_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const utc_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday &__wd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_indexed &__wdi)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_last &__wdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month &__ym)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day &__ymd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day_last &__ymdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday &__ymwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday_last &__ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const zoned_time< _Duration, _TimeZonePtr > &__t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const duration< _Rep, _Period > &__d)`
- `template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>`  
`bool std::chrono::operator== (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2`  
`> &__y)`
- `constexpr weekday_last std::chrono::weekday::operator[] (last_spec) const noexcept`
- `constexpr weekday_indexed std::chrono::weekday::operator[] (unsigned __index) const noexcept`

- `template<typename _CharT, __detail::__parsable<_CharT> _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable<_CharT, _Traits, _StrT> _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic\_string<_CharT, _Traits, _Alloc> & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable<_CharT, _Traits, _StrT, minutes> _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic\_string<_CharT, _Traits, _Alloc> & __abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<_CharT, _Traits, _StrT, minutes> _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable<_CharT, _Traits> _Parsable>`  
`auto std::chrono::parse (const basic\_string<_CharT, _Traits, _Alloc> & __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable<_CharT, _Traits, _StrT> _Parsable>`  
`auto std::chrono::parse (const basic\_string<_CharT, _Traits, _Alloc> & __fmt, _Parsable & __tp, basic\_string<_CharT, _Traits, _Alloc> & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable<_CharT, _Traits, _StrT, minutes> _Parsable>`  
`auto std::chrono::parse (const basic\_string<_CharT, _Traits, _Alloc> & __fmt, _Parsable & __tp, basic\_string<_CharT, _Traits, _Alloc> & __abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<_CharT, _Traits, _StrT, minutes> _Parsable>`  
`auto std::chrono::parse (const basic\_string<_CharT, _Traits, _Alloc> & __fmt, _Parsable & __tp, minutes & __offset)`
- `const tzdb & std::chrono::reload_tzdb ()`
- `string std::chrono::remote_version ()`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr enable\_if\_t< __and< __is_duration<_ToDur>, __not< treat\_as\_floating\_point<typename _ToDur::rep>>>::value, _ToDur> std::chrono::round (const duration<_Rep, _Period> & __d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable\_if\_t< __is_duration_v<_ToDur> && !treat\_as\_floating\_point\_v<typename _ToDur::rep>, time\_point<_Clock, _ToDur>> std::chrono::round (const time\_point<_Clock, _Dur> & __tp)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr __enable_if_t< __is_duration<_ToDur>::value, time\_point<_Clock, _ToDur>> std::chrono::time_point_cast (const time\_point<_Clock, _Dur> & __t)`
- `std::chrono::zoned_time () -> zoned_time<seconds>`
- `template<typename _TimeZonePtrOrName>`  
`std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time<seconds, __time_zone_representation<_TimeZonePtrOrName>>`
- `template<typename _TimeZonePtrOrName, typename _Duration>`  
`std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time<_Duration>, choose=choose::earliest) -> zoned_time<common\_type\_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName>>`
- `template<typename _TimeZonePtrOrName, typename _Duration>`  
`std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time<_Duration>) -> zoned_time<common\_type\_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName>>`
- `template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>`  
`std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time<_Duration, _TimeZonePtr2>, choose=choose::earliest) -> zoned_time<common\_type\_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName>>`

- `template<typename _Duration>`  
`std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >`  
`>`

## Variables

- `constexpr month std::chrono::April`
- `constexpr month std::chrono::August`
- `constexpr month std::chrono::December`
- `constexpr month std::chrono::February`
- `constexpr weekday std::chrono::Friday`
- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v`
- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v< _Tp >`
- `template<> constexpr bool std::chrono::is_clock_v< file_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< gps_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< steady_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< system_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< tai_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< utc_clock >`
- `constexpr month std::chrono::January`
- `constexpr month std::chrono::July`
- `constexpr month std::chrono::June`
- `constexpr last_spec std::chrono::last`
- `constexpr month std::chrono::March`
- `constexpr month std::chrono::May`
- `constexpr weekday std::chrono::Monday`
- `constexpr month std::chrono::November`
- `constexpr month std::chrono::October`
- `constexpr weekday std::chrono::Saturday`
- `constexpr month std::chrono::September`
- `constexpr weekday std::chrono::Sunday`
- `constexpr weekday std::chrono::Thursday`
- `template<typename _Rep>`  
`constexpr bool std::chrono::treat_as_floating_point_v`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< float >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< int >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long long >`
- `constexpr weekday std::chrono::Tuesday`
- `constexpr weekday std::chrono::Wednesday`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator-`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

- `template<typename _Rep1, typename _Rep2, typename _Period>`  
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s,`  
`const duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period >`  
`std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period >`  
`std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator%`  
`(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, __`  
`Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr auto std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >`  
`std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__`  
`__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

- `template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>`  
`constexpr auto std::chrono::operator<=> (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _↵`  
`_Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator< (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator<= (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _↵`  
`_Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator> (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator>= (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _↵`  
`_Clock, _Dur2 > &__rhs)`

### 3.16.9.1 Detailed Description

Classes and functions for time.

Since

C++11

### 3.16.9.2 Typedef Documentation

#### **days**

```
using std::chrono::days
days
```

#### **high\_resolution\_clock**

```
using std::chrono::high_resolution_clock
```

Highest-resolution clock.

This is the clock "with the shortest tick period." Alias to `std::system_clock` until higher-than-nanosecond definitions become feasible.

#### **hours**

```
using std::chrono::hours
hours
```

#### **microseconds**

```
using std::chrono::microseconds
microseconds
```

#### **milliseconds**

```
using std::chrono::milliseconds
milliseconds
```

#### **minutes**

```
using std::chrono::minutes
minutes
```

**months**

```
using std::chrono::months
months
```

**nanoseconds**

```
using std::chrono::nanoseconds
nanoseconds
```

**seconds**

```
using std::chrono::seconds
seconds
```

**weeks**

```
using std::chrono::weeks
weeks
```

**years**

```
using std::chrono::years
years
```

**3.16.9.3 Function Documentation****abs()**

```
template<typename _Rep, typename _Period>
enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (
 duration< _Rep, _Period > __d) [nodiscard], [constexpr]
```

The absolute (non-negative) value of a duration.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| $\leftarrow$<br>_d | A duration with a signed rep type. |
|--------------------|------------------------------------|

**Returns**

A duration of the same type as the argument, with value  $|d|$ .

**Since**

C++17

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

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
 const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round up.

If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.



**Template Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>_ToDur</code> | The result type must be a duration. |
|---------------------|-------------------------------------|

**Parameters**

|                  |             |
|------------------|-------------|
| <code>__d</code> | A duration. |
|------------------|-------------|

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [duration\\_cast\(\)](#).

Referenced by [ceil\(\)](#).

**ceil() [2/2]**

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (
 const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round up.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

**Template Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>_ToDur</code> | The <code>duration</code> type to use for the result. |
|---------------------|-------------------------------------------------------|

**Parameters**

|                   |               |
|-------------------|---------------|
| <code>__tp</code> | A time point. |
|-------------------|---------------|

**Returns**

The value of `__tp` rounded up to the precision of `_ToDur`.

**Since**

C++17

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

**clock\_cast()**

```
template<typename _DestClock, typename _SourceClock, typename _Duration>
requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs↔
_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock,
_Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::__↔
_clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
auto std::chrono::clock_cast (
 const time_point< _SourceClock, _Duration > & __t) [inline], [nodiscard]
```

Convert a time point to a different clock.

**duration\_cast()**

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::duration_cast (
 const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type ToDur.

If the duration cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

**Template Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>_ToDur</code> | The result type must be a duration. |
|---------------------|-------------------------------------|

**Parameters**

|                         |             |
|-------------------------|-------------|
| <code>__↔<br/>_d</code> | A duration. |
|-------------------------|-------------|

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++11

Referenced by [ceil\(\)](#), [floor\(\)](#), and [time\\_point\\_cast\(\)](#).

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

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::floor (
 const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type ToDur and round down.

If the duration cannot be represented exactly in the result type, returns the closest value that is less than the argument.

**Template Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>_ToDur</code> | The result type must be a duration. |
|---------------------|-------------------------------------|

**Parameters**

|                                                                 |             |
|-----------------------------------------------------------------|-------------|
| <a href="#"><math>\leftrightarrow</math></a><br><code>_d</code> | A duration. |
|-----------------------------------------------------------------|-------------|

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

References [duration\\_cast\(\)](#).

Referenced by [floor\(\)](#), and [round\(\)](#).

**floor()** [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (
 const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round down.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is less than the argument.

**Template Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>_ToDur</code> | The <code>duration</code> type to use for the result. |
|---------------------|-------------------------------------------------------|

**Parameters**

|                                                                  |               |
|------------------------------------------------------------------|---------------|
| <a href="#"><math>\leftrightarrow</math></a><br><code>_tp</code> | A time point. |
|------------------------------------------------------------------|---------------|

**Returns**

The value of `__tp` rounded down to the precision of `_ToDur`.

**Since**

C++17

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

**is\_am()**

```
bool std::chrono::is_am (
 const hours & __h) [constexpr], [noexcept]
```

True if a `chrono::hours` value represents a time before midday.

References [is\\_am\(\)](#).

Referenced by [is\\_am\(\)](#).

**is\_pm()**

```
bool std::chrono::is_pm (
 const hours & __h) [constexpr, [noexcept]
```

True if a `chrono::hours` value represents a time after midday.

References [is\\_pm\(\)](#).

Referenced by [is\\_pm\(\)](#).

**local\_time\_format()**

```
template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (
 local_time< _Duration > __time,
 const string * __abbrev = nullptr,
 const seconds * __offset_sec = nullptr) [inline]
```

Return an object that associates timezone info with a local time.

A `chrono::local_time` object has no timezone associated with it. This function creates an object that allows formatting a `local_time` as though it refers to a timezone with the given abbreviated name and offset from UTC.

Since

C++20

**make12()**

```
hours std::chrono::make12 (
 const hours & __h) [constexpr, [noexcept]
```

Convert a 24-hour time into a 12-hour time.

References [make12\(\)](#).

Referenced by [make12\(\)](#).

**make24()**

```
hours std::chrono::make24 (
 const hours & __h,
 bool __is_pm) [constexpr, [noexcept]
```

Convert a 12-hour time and p.m. flag into a 24-hour time.

References [make24\(\)](#).

Referenced by [make24\(\)](#).

**operator""d()**

```
chrono::day std::literals::chrono_literals::operator""d (
 unsigned long long __d) [constexpr, [noexcept]
```

Literal suffix for creating `chrono::day` objects.

Since

C++20

**operator""h() [1/2]**

```
template<char... _Digits>
chrono::hours std::literals::chrono_literals::operator""h () [constexpr]
```

Literal suffix for durations of type `std::chrono::hours`.

**operator""h() [2/2]**

```
chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (
 long double __hours) [constexpr]
```

Literal suffix for durations representing non-integer hours.

**operator""min() [1/2]**

```
template<char... _Digits>
chrono::minutes std::literals::chrono_literals::operator""min () [constexpr]
```

Literal suffix for durations of type `std::chrono::minutes`.

**operator""min() [2/2]**

```
chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (
 long double __mins) [constexpr]
```

Literal suffix for durations representing non-integer minutes.

**operator""ms() [1/2]**

```
template<char... _Digits>
chrono::milliseconds std::literals::chrono_literals::operator""ms () [constexpr]
```

Literal suffix for durations of type `std::chrono::milliseconds`.

**operator""ms() [2/2]**

```
chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (
 long double __msecs) [constexpr]
```

Literal suffix for durations representing non-integer milliseconds.

**operator""ns() [1/2]**

```
template<char... _Digits>
chrono::nanoseconds std::literals::chrono_literals::operator""ns () [constexpr]
```

Literal suffix for durations of type `std::chrono::nanoseconds`.

**operator""ns() [2/2]**

```
chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (
 long double __nsecs) [constexpr]
```

Literal suffix for durations representing non-integer nanoseconds.

**operator""s() [1/2]**

```
template<char... _Digits>
chrono::seconds std::literals::chrono_literals::operator""s () [constexpr]
```

Literal suffix for durations of type `std::chrono::seconds`.

**operator""s() [2/2]**

```
chrono::duration< long double > std::literals::chrono_literals::operator""s (
 long double __secs) [constexpr]
```

Literal suffix for durations representing non-integer seconds.

**operator""us() [1/2]**

```
template<char... _Digits>
chrono::microseconds std::literals::chrono_literals::operator""us () [constexpr]
Literal suffix for durations of type std::chrono::microseconds.
```

**operator""us() [2/2]**

```
chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (
 long double __usecs) [constexpr]
Literal suffix for durations representing non-integer microseconds.
```

**operator""y()**

```
chrono::year std::literals::chrono_literals::operator""y (
 unsigned long long __y) [constexpr], [noexcept]
Literal suffix for creating chrono::year objects.
```

Since

C++20

**operator%() [1/2]**

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::←
operator% (
 const duration< _Rep1, _Period > & __d,
 const _Rep2 & __s) [constexpr]
```

Arithmetic operators for chrono::duration

**operator%() [2/2]**

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::←
operator% (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Arithmetic operators for chrono::duration

**operator\*() [1/2]**

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (
 const duration< _Rep1, _Period > & __d,
 const _Rep2 & __s) [related]
```

Arithmetic operators for chrono::duration

**operator\*() [2/2]**

```
template<typename _Rep1, typename _Rep2, typename _Period>
duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (
 const _Rep1 & __s,
 const duration< _Rep2, _Period > & __d) [constexpr]
```

Arithmetic operators for chrono::duration

**operator+()** [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [related]
```

The sum of two durations.

**operator+()** [2/3]

```
template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::chrono::operator+ (
 const duration< _Rep1, _Period1 > & __lhs,
 const time_point< _Clock, _Dur2 > & __rhs) [constexpr]
```

Adjust a time point forwards by the given duration.

**operator+()** [3/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (
 const time_point< _Clock, _Dur1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [related]
```

Adjust a time point forwards by the given duration.

**operator-()** [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator- (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

The difference between two durations.

**operator-()** [2/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (
 const time_point< _Clock, _Dur1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Adjust a time point backwards by the given duration.

**operator-()** [3/3]

```
template<typename _Clock, typename _Dur1, typename _Dur2>
common_type< _Dur1, _Dur2 >::type std::chrono::operator- (
 const time_point< _Clock, _Dur1 > & __lhs,
 const time_point< _Clock, _Dur2 > & __rhs) [constexpr]
```

The difference between two time points (as a duration).

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

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::
```

```
operator/ (
 const duration< _Rep1, _Period > & __d,
 const _Rep2 & __s) [constexpr]
```

Arithmetic operators for chrono::duration

### operator/() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Arithmetic operators for chrono::duration

### operator<()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator< (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

### operator<<() [1/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const local_info & __li)
```

Writes a local\_info object to an ostream in an unspecified format.

### operator<<() [2/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const sys_info & __i)
```

Writes a sys\_info object to an ostream in an unspecified format.

### operator<<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const duration< _Rep, _Period > & __d) [inline]
```

Write a chrono::duration to an ostream.

Since

C++20

References [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::imbue\(\)](#), [std::move\(\)](#), [operator<<\(\)](#), and [std::ios\\_base::precision\(\)](#).

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

### operator<=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator<= (
```



```
const duration< _Rep1, _Period1 > & __lhs,
const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

#### operator<=>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
auto std::chrono::operator<=() (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

#### operator==(())

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool operator==(
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [related]
```

Comparisons for chrono::duration

#### operator>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator> (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

#### operator>=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator>= (
 const duration< _Rep1, _Period1 > & __lhs,
 const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Comparisons for chrono::duration

#### round() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
enable_if_t< __and< __is_duration< _ToDur >, __not< treat_as_floating_point< typename _ToDur::rep > > >::value, _ToDur > std::chrono::round (
 const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type ToDur and round to the closest value.

If the duration cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

#### Template Parameters

|                     |                                                                        |
|---------------------|------------------------------------------------------------------------|
| <code>_ToDur</code> | The result type must be a duration with a non-floating-point rep type. |
|---------------------|------------------------------------------------------------------------|

#### Parameters

|                  |             |
|------------------|-------------|
| <code>__d</code> | A duration. |
|------------------|-------------|

**Returns**

The value of `__d` converted to type `_ToDur`.

**Since**

C++17

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

Referenced by [round\(\)](#).

**round()** [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,
time_point< _Clock, _ToDur > > std::chrono::round (
 const time_point< _Clock, _Dur > & __tp) [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round to the closest value.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

**Template Parameters**

|                     |                                                                                                                   |
|---------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>_ToDur</code> | The <code>duration</code> type to use for the result, which must have a non-floating-point <code>rep</code> type. |
|---------------------|-------------------------------------------------------------------------------------------------------------------|

**Parameters**

|                   |               |
|-------------------|---------------|
| <code>__tp</code> | A time point. |
|-------------------|---------------|

**Returns**

The value of `__tp` rounded to the precision `_ToDur`.

**Since**

C++17

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

**time\_point\_cast()**

```
template<typename _ToDur, typename _Clock, typename _Dur>
__enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::
time_point_cast (
 const time_point< _Clock, _Dur > & __t) [nodiscard], [constexpr]
```

Convert a `time_point` to use `duration` type `ToDur`.

The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

**Template Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>_ToDur</code> | The <code>duration</code> type to use for the result. |
|---------------------|-------------------------------------------------------|

**Parameters**

|                |               |
|----------------|---------------|
| <code>↵</code> | A time point. |
| <code>↵</code> |               |
| <code>↵</code> |               |
| <code>↵</code> |               |
| <code>t</code> |               |

**Returns**

The value of `__t` converted to use type `_ToDur`.

**Since**

C++11

References [duration\\_cast\(\)](#).

## 4 Namespace Documentation

### 4.1 `__gnu_cxx` Namespace Reference

**Namespaces**

- namespace [\\_\\_detail](#)
- namespace [typelist](#)

**Classes**

- struct [\\_\\_alloc\\_traits](#)
- struct [\\_\\_common\\_pool\\_policy](#)
- class [\\_\\_mt\\_alloc](#)
- class [\\_\\_mt\\_alloc\\_base](#)
- struct [\\_\\_per\\_type\\_pool\\_policy](#)
- class [\\_\\_pool](#)
- class [\\_\\_pool< false >](#)
- class [\\_\\_pool< true >](#)
- class [\\_\\_pool\\_alloc](#)
- class [\\_\\_pool\\_alloc\\_base](#)
- struct [\\_\\_pool\\_base](#)
- class [\\_\\_rc\\_string\\_base](#)
- class [\\_\\_scoped\\_lock](#)
- class [\\_\\_versa\\_string](#)
- struct [\\_Caster](#)
- struct [\\_Char\\_types](#)
- class [\\_ExtPtr\\_allocator](#)
- struct [\\_Invalid\\_type](#)

- class [\\_Pointer\\_adapter](#)
- class [\\_Relative\\_pointer\\_impl](#)
- class [\\_Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_Std\\_pointer\\_impl](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Unqualified\\_type](#)
- class [allocator](#)
- struct [annotate\\_base](#)
- class [binary\\_compose](#)
- class [bitmap\\_allocator](#)
- struct [char\\_traits](#)
- struct [character](#)
- struct [condition\\_base](#)
- struct [constant\\_binary\\_fun](#)
- struct [constant\\_unary\\_fun](#)
- struct [constant\\_void\\_fun](#)
- class [debug\\_allocator](#)
- class [enc\\_filebuf](#)
- struct [encoding\\_char\\_traits](#)
- class [encoding\\_state](#)
- struct [equal\\_to](#)
- struct [forced\\_error](#)
- class [free\\_list](#)
- class [hash\\_map](#)
- class [hash\\_multimap](#)
- class [hash\\_multiset](#)
- class [hash\\_set](#)
- struct [limit\\_condition](#)
- class [malloc\\_allocator](#)
- class [new\\_allocator](#)
- struct [pair](#)
- struct [project1st](#)
- struct [project2nd](#)
- struct [random\\_condition](#)
- struct [rb\\_tree](#)
- class [recursive\\_init\\_error](#)
- class [rope](#)
- struct [select1st](#)
- struct [select2nd](#)
- class [slist](#)
- class [stdio\\_filebuf](#)
- class [stdio\\_sync\\_filebuf](#)
- class [subtractive\\_rng](#)
- struct [temporary\\_buffer](#)
- class [throw\\_allocator\\_base](#)
- struct [throw\\_allocator\\_limit](#)
- struct [throw\\_allocator\\_random](#)
- struct [throw\\_value\\_base](#)
- struct [throw\\_value\\_limit](#)
- struct [throw\\_value\\_random](#)
- class [unary\\_compose](#)

## Typedefs

- typedef void(\* **\_\_destroy\_handler**) (void \*)
- template<typename \_Tp>  
using **\_\_int\_traits**
- typedef **\_\_versa\_string**< char, **std::char\_traits**< char >, **std::allocator**< char >, **\_\_rc\_string\_base** > **\_\_rc\_string**
- typedef **\_\_vstring** **\_\_sso\_string**
- typedef **\_\_versa\_string**< char16\_t, **std::char\_traits**< char16\_t >, **std::allocator**< char16\_t >, **\_\_rc\_string\_base** > **\_\_u16rc\_string**
- typedef **\_\_u16vstring** **\_\_u16sso\_string**
- typedef **\_\_versa\_string**< char16\_t > **\_\_u16vstring**
- typedef **\_\_versa\_string**< char32\_t, **std::char\_traits**< char32\_t >, **std::allocator**< char32\_t >, **\_\_rc\_string\_base** > **\_\_u32rc\_string**
- typedef **\_\_u32vstring** **\_\_u32sso\_string**
- typedef **\_\_versa\_string**< char32\_t > **\_\_u32vstring**
- typedef **\_\_versa\_string**< char > **\_\_vstring**
- typedef **\_\_versa\_string**< wchar\_t, **std::char\_traits**< wchar\_t >, **std::allocator**< wchar\_t >, **\_\_rc\_string\_base** > **\_\_wrc\_string**
- typedef **\_\_wvstring** **\_\_wsso\_string**
- typedef **\_\_versa\_string**< wchar\_t > **\_\_wvstring**
- typedef **rope**< char > **crope**
- typedef **\_\_SIZE\_TYPE\_\_** **\_\_size\_t**
- typedef **rope**< wchar\_t > **wrope**

## Enumerations

- enum { **\_S\_num\_primes** }
- enum **\_Lock\_policy** { **\_S\_single** , **\_S\_mutex** , **\_S\_atomic** }

## Functions

- void **\_\_atomic\_add** (volatile **\_Atomic\_word** \* \_\_mem, int \_\_val)
- void **\_\_atomic\_add\_dispatch** (**\_Atomic\_word** \* \_\_mem, int \_\_val)
- void **\_\_atomic\_add\_single** (**\_Atomic\_word** \* \_\_mem, int \_\_val)
- template<class \_Tp>  
void **\_\_aux\_require\_boolean\_expr** (const \_Tp & \_\_t)
- template<typename \_ToType, typename \_FromType>  
**\_ToType** **\_\_const\_pointer\_cast** (**\_FromType** \* \_\_arg)
- template<typename \_ToType, typename \_FromType>  
**\_ToType** **\_\_const\_pointer\_cast** (const **\_FromType** & \_\_arg)
- template<typename \_InputIterator, typename \_Size, typename \_OutputIterator>  
**std::pair**< **\_InputIterator**, **\_OutputIterator** > **\_\_copy\_n** (**\_InputIterator** \_\_first, **\_Size** \_\_count, **\_OutputIterator** \_\_result, **std::input\_iterator\_tag**)
- template<typename \_RAIterator, typename \_Size, typename \_OutputIterator>  
**std::pair**< **\_RAIterator**, **\_OutputIterator** > **\_\_copy\_n** (**\_RAIterator** \_\_first, **\_Size** \_\_count, **\_OutputIterator** \_\_result, **std::random\_access\_iterator\_tag**)
- template<typename \_InputIterator, typename \_Distance>  
void **\_\_distance** (**\_InputIterator** \_\_first, **\_InputIterator** \_\_last, **\_Distance** & \_\_n, **std::input\_iterator\_tag**)
- template<typename \_RandomAccessIterator, typename \_Distance>  
void **\_\_distance** (**\_RandomAccessIterator** \_\_first, **\_RandomAccessIterator** \_\_last, **\_Distance** & \_\_n, **std::random\_access\_iterator\_tag**)
- template<typename \_ToType, typename \_FromType>  
**\_ToType** **\_\_dynamic\_pointer\_cast** (**\_FromType** \* \_\_arg)

- `template<typename _ToType, typename _FromType>`  
`_ToType __dynamic_pointer_cast (const _FromType &__arg)`
- `void __error_type_must_be_a_signed_integer_type ()`
- `void __error_type_must_be_an_integer_type ()`
- `void __error_type_must_be_an_unsigned_integer_type ()`
- `_Atomic_word __exchange_and_add (volatile _Atomic_word *__mem, int __val)`
- `_Atomic_word __exchange_and_add_dispatch (_Atomic_word *__mem, int __val)`
- `_Atomic_word __exchange_and_add_single (_Atomic_word *__mem, int __val)`
- `template<class _Concept>`  
`constexpr void __function_requires ()`
- `template<typename _Type>`  
`constexpr bool __is_null_pointer (_Type *__ptr)`
- `template<typename _Type>`  
`constexpr bool __is_null_pointer (_Type)`
- `constexpr bool __is_null_pointer (std::nullptr_t)`
- `bool __is_single_threaded () noexcept`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int __lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int __lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `int __lexicographical_compare_3way (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)`
- `template<typename _Tp>`  
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare>`  
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference __mutable_reference_at (crope &__c, std::size_t __i)`
- `template<typename _Tp, typename _Integer>`  
`_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>`  
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>`  
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __reinterpret_pointer_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base *__head, const _Slist_node_base *__node)`
- `const _Slist_node_base * __slist_previous (const _Slist_node_base *__head, const _Slist_node_base *__node)`
- `_Slist_node_base * __slist_reverse (_Slist_node_base *__node)`
- `std::size_t __slist_size (_Slist_node_base *__node)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__head)`

- `template<typename _ToType, typename _FromType>`  
`_ToType \_\_static\_pointer\_cast (_FromType * __arg)`
- `template<typename _ToType, typename _FromType>`  
`_ToType \_\_static\_pointer\_cast (const _FromType & __arg)`
- `size_t \_\_stl\_hash\_string (const char * __s)`
- `unsigned long \_\_stl\_next\_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`  
`_Ret \_\_stoa (_TRet(* __convf)(const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str,`  
`std::size_t * __idx, _Base... __base)`
- `void \_\_throw\_concurrency\_lock\_error ()`
- `void \_\_throw\_concurrency\_unlock\_error ()`
- `void \_\_throw\_forced\_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type>`  
`_String \_\_to\_xstring (int(* __convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n,`  
`const _CharT * __fmt,...)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter`  
`__result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n (_InputIter __first, _Size __count, _ForwardIter`  
`__result, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>`  
`std::pair< _RandomAccessIter, _ForwardIter > \_\_uninitialized\_copy\_n (_RandomAccessIter __first, _Size __count,`  
`_ForwardIter __result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>`  
`std::pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n\_a (_InputIter __first, _Size __count, _ForwardIter`  
`__result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>`  
`std::pair< _InputIter, _ForwardIter > \_\_uninitialized\_copy\_n\_a (_InputIter __first, _Size __count, _ForwardIter`  
`__result, std::allocator< _Tp >)`
- `void \_\_verbose\_terminate\_handler ()`
- `std::size_t \_\_Bit\_scan\_forward (std::size_t __num)`
- `template<typename _ForwardIterator, typename _Allocator>`  
`void \_\_Destroy\_const (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp>`  
`void \_\_Destroy\_const (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)`
- `template<class _CharT, class _Traits>`  
`void \_\_Rope\_fill (std::basic\_ostream< _CharT, _Traits > & __o, std::size_t __n)`
- `template<class _CharT>`  
`bool \_\_Rope\_is\_simple (_CharT *)`
- `bool \_\_Rope\_is\_simple (char *)`
- `bool \_\_Rope\_is\_simple (wchar_t *)`
- `template<class _Rope_iterator>`  
`void \_\_Rope\_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT>`  
`void \_\_S\_cond\_store\_eos (_CharT &)`
- `void \_\_S\_cond\_store\_eos (char & __c)`
- `void \_\_S\_cond\_store\_eos (wchar_t & __c)`
- `template<class _CharT>`  
`_CharT \_\_S\_eos (_CharT *)`
- `template<class _CharT>`  
`bool \_\_S\_is\_basic\_char\_type (_CharT *)`

- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT>`  
`bool _S_is_one_byte_char_type (_CharT *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type airy_ai (_Tp __x)`
- `float airy_aif (float __x)`
- `long double airy_ail (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type airy_bi (_Tp __x)`
- `float airy_bif (float __x)`
- `long double airy_bil (long double __x)`
- `template<class _Operation1, class _Operation2>`  
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3>`  
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyberg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hybergf (float __a, float __c, float __x)`
- `long double conf_hybergl (long double __a, long double __c, long double __x)`
- `template<class _Result>`  
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result>`  
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result>`  
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size>`  
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size>`  
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator, typename _Distance>`  
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<class _Tp>`  
`_Tp identity_element (std::multiplies< _Tp >)`
- `template<class _Tp>`  
`_Tp identity_element (std::plus< _Tp >)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter>`  
`constexpr bool is_heap (_RAIter, _RAIter)`
- `template<typename _Filter>`  
`constexpr bool is_sorted (_Filter, _Filter)`



- `template<typename _InputIterator1, typename _InputIterator2>`  
`int lexicographical\_compare\_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::const\_mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun1 (_Ret(_Tp::*_f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun1 (_Ret(_Tp::*_f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::const\_mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun1\_ref (_Ret(_Tp::*_f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg>`  
`std::mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun1\_ref (_Ret(_Tp::*_f)(_Arg))`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (_Tp1 __lhs, const Pointer\_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const versa\_string< _CharT,  
_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const versa\_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator!= (const _CharT *__lhs, const versa\_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator!= (const Pointer\_adapter< _Tp > &__lhs, const Pointer\_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool operator!= (const Pointer\_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const Pointer\_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const Pointer\_adapter< _Tp1 > &__lhs, const Pointer\_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const Rope\_char\_ptr\_proxy< _CharT, _Alloc > &__x, const Rope\_char\_ptr\_proxy< _CharT,  
_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const Rope\_const\_iterator< _CharT, _Alloc > &__x, const Rope\_const\_iterator< _CharT,  
_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const Rope\_iterator< _CharT, _Alloc > &__x, const Rope\_iterator< _CharT, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`bool operator!= (const hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash\_map< _Key,  
_Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>`  
`bool operator!= (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash\_multimap<  
_Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator!= (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_multiset<  
_Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator!= (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash\_set< _Value,  
_HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF,  
_Ex, _Eq, _All > &__ht2)`
- `template<class _CharT, class _Alloc>`  
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`

- `template<class _Tp, class _Alloc>`  
`bool operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Tp>`  
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( __versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( _CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ ( _CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`

- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__`  
`__right)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base<`  
`_Cond > &__b)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT,`  
`_Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< __`  
`CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__`  
`x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc >`  
`&__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<typename _Cond>`  
`throw_value_base< _Cond > operator- (const throw_value_base< _Cond > &__a, const throw_value_base<`  
`_Cond > &__b)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`  
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`  
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Value, typename _Int, typename _St>`  
`bool operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`

- `template<class _Tp, class _Alloc>`  
`bool operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Cond>`  
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _StoreT>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const logistic_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`rice_distribution< _RealType > &__x)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= ( _Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`  
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`  
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator<= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== ( _Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<size_t _Dimen, typename _RealType>`  
`bool operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen,`  
`_RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1,`  
`__msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const`  
`__gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1,`  
`__msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<typename _Tp, typename _Poolp>`  
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp>`  
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`  
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator== (const _CharT * __lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _Tp>`  
`bool operator== (const \_Pointer\_adapter< _Tp > & __lhs, const \_Pointer\_adapter< _Tp > & __rhs)`
- `template<typename _Tp>`  
`bool operator== (const \_Pointer\_adapter< _Tp > & __lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > & __lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const \_Pointer\_adapter< _Tp1 > & __lhs, const \_Pointer\_adapter< _Tp2 > & __rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const \_Rope\_char\_ptr\_proxy< _CharT, _Alloc > & __x, const \_Rope\_char\_ptr\_proxy< _CharT, _Alloc > & __y)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const \_Rope\_const\_iterator< _CharT, _Alloc > & __x, const \_Rope\_const\_iterator< _CharT, _Alloc > & __y)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const \_Rope\_iterator< _CharT, _Alloc > & __x, const \_Rope\_iterator< _CharT, _Alloc > & __y)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const bitmap\_allocator< _Tp1 > &, const bitmap\_allocator< _Tp2 > &) throw ()`
- `template<typename _Value, typename _Int, typename _St>`  
`bool operator== (const character< _Value, _Int, _St > & lhs, const character< _Value, _Int, _St > & rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`bool operator== (const hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm1, const hash\_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>`  
`bool operator== (const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > & __hm1, const hash\_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > & __hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator== (const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs1, const hash\_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator== (const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > & __hs1, const hash\_set< _Value, _HashFcn, _EqualKey, _Alloc > & __hs2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>`  
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > & __ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > & __ht2)`
- `template<class _CharT, class _Alloc>`  
`bool operator== (const rope< _CharT, _Alloc > & __left, const rope< _CharT, _Alloc > & __right)`
- `template<class _Tp, class _Alloc>`  
`bool operator== (const slist< _Tp, _Alloc > & __SL1, const slist< _Tp, _Alloc > & __SL2)`
- `template<typename _Tp, typename _Cond>`  
`bool operator== (const throw\_allocator\_base< _Tp, _Cond > &, const throw\_allocator\_base< _Tp, _Cond > &)`
- `template<typename _Cond>`  
`bool operator== (const throw\_value\_base< _Cond > & __a, const throw\_value\_base< _Cond > & __b)`
- `template<typename _Tp>`  
`bool operator== (int __lhs, const \_Pointer\_adapter< _Tp > & __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> ( _Tp1 __lhs, const \_Pointer\_adapter< _Tp2 > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const \_\_versa\_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`



- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator> (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp>`  
`bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _Tp, class _Alloc>`  
`bool operator>= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`

- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::triangular_distribution< _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::von_mises_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, arcsine_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, hoyt_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, k_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, logistic_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, nakagami_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, pareto_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, rice_distribution< _RealType > & __x)`
- `template<typename _Tp, typename _Integer>`  
`_Tp power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator & __rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`



- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`  
`_OutputIterator random_sample_n(_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const`  
`_Distance __n, _RandomNumberGenerator &__rand)`
- `void rotate(_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`
- `double stod(const __vstring &__str, std::size_t * __idx=0)`
- `double stod(const __wvstring &__str, std::size_t * __idx=0)`
- `float stof(const __vstring &__str, std::size_t * __idx=0)`
- `float stof(const __wvstring &__str, std::size_t * __idx=0)`
- `int stoi(const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `int stoi(const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `long stol(const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `long stol(const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `long double stold(const __vstring &__str, std::size_t * __idx=0)`
- `long double stold(const __wvstring &__str, std::size_t * __idx=0)`
- `long long stoll(const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `long long stoll(const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long stoul(const __vstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long stoul(const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `unsigned long long stoull(const __vstring &__str, std::size_t * __idx, int __base=10)`
- `unsigned long long stoull(const __wvstring &__str, std::size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap(__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc,`  
`_Base > &__rhs)`
- `template<typename _Tp>`  
`void swap(_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`
- `template<class _CharT, class __Alloc>`  
`void swap(_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`void swap(hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn,`  
`_EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>`  
`void swap(hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp,`  
`_HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void swap(hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn,`  
`_EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void swap(hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey,`  
`_Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>`  
`void swap(hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract,`  
`_EqKey, _All > &__ht2)`
- `template<class _CharT, class __Alloc>`  
`void swap(rope< _CharT, __Alloc > &__x, rope< _CharT, __Alloc > &__y)`
- `template<class _Tp, class __Alloc>`  
`void swap(slist< _Tp, __Alloc > &__x, slist< _Tp, __Alloc > &__y)`
- `template<typename _Cond>`  
`void swap(throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `__vstring to_string(double __val)`
- `__vstring to_string(float __val)`

- `__vstring to_string` (int \_\_val)
- `__vstring to_string` (long \_\_val)
- `__vstring to_string` (long double \_\_val)
- `__vstring to_string` (long long \_\_val)
- `__vstring to_string` (unsigned \_\_val)
- `__vstring to_string` (unsigned long \_\_val)
- `__vstring to_string` (unsigned long long \_\_val)
- `__wvstring to_wstring` (double \_\_val)
- `__wvstring to_wstring` (float \_\_val)
- `__wvstring to_wstring` (int \_\_val)
- `__wvstring to_wstring` (long \_\_val)
- `__wvstring to_wstring` (long double \_\_val)
- `__wvstring to_wstring` (long long \_\_val)
- `__wvstring to_wstring` (unsigned \_\_val)
- `__wvstring to_wstring` (unsigned long \_\_val)
- `__wvstring to_wstring` (unsigned long long \_\_val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
`std::pair< _InputIter, _ForwardIter > uninitialized_copy_n ( _InputIter __first, _Size __count, _ForwardIter __`  
`result)`

## Variables

- `const _Lock_policy __default_lock_policy`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__gamma_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__ln_10`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__ln_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__ln_3`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__log10_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__log2_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__one_div_e`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__one_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__one_div_root_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__phi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_half`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_quarter`

- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__pi_third`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_3`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_5`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_7`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__root_pi_div_2`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__two_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __math_constants< _RealType >::__two_div_root_pi`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__digits10`
- `template<typename _Value>`  
`const bool __numeric_traits_floating< _Value >::__is_signed`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__max_digits10`
- `template<typename _Value>`  
`const int __numeric_traits_floating< _Value >::__max_exponent10`
- `template<typename _Value>`  
`const int __numeric_traits_integer< _Value >::__digits`
- `template<typename _Value>`  
`const bool __numeric_traits_integer< _Value >::__is_signed`
- `template<typename _Value>`  
`const _Value __numeric_traits_integer< _Value >::__max`
- `template<typename _Value>`  
`const _Value __numeric_traits_integer< _Value >::__min`
- `template<typename _Tp>`  
`_Atomic_word __pool_alloc< _Tp >::__S_force_new`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`__rc_string_base< _CharT, _Traits, _Alloc >::__Rep_empty __rc_string_base< _CharT, _Traits, _Alloc >::__S_empty_rep`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __versa_string< _CharT, _Traits, _Alloc, _Base >::npos`
- `template<typename _PrimeType>`  
`const _PrimeType __Hashtable_prime_list< _PrimeType >::__stl_prime_list [_S_num_primes]`
- `template<typename _Tp>`  
`std::size_t bitmap_allocator< _Tp >::__S_block_size`
- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::__BPVector::size_type bitmap_allocator< _Tp >::__S_last_dealloc_index`
- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::__BPVector bitmap_allocator< _Tp >::__S_mem_blocks`
- `template<typename _Tp>`  
`bitmap_allocator< _Tp >::__mutex_type bitmap_allocator< _Tp >::__S_mut`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

- `template<class _CharT, class _Alloc>`  
`_CharT rope< _CharT, _Alloc >::_S_empty_c_str [1]`
- `template<class _CharT, class _Alloc>`  
`const unsigned long rope< _CharT, _Alloc >::_S_min_len [int(__detail::_S_max_rope_depth)+1]`
- `template<class _CharT, class _Alloc>`  
`const rope< _CharT, _Alloc >::size_type rope< _CharT, _Alloc >::npos`

#### 4.1.1 Detailed Description

GNU extensions for public use.

#### 4.1.2 Typedef Documentation

##### `__int_traits`

```
template<typename _Tp>
using __gnu_cxx::__int_traits
Convenience alias for __numeric_traits<integer-type>.
```

#### 4.1.3 Function Documentation

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

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
 _FromType * __arg) [inline]
```

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

##### `__static_pointer_cast()` [2/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
 const _FromType & __arg) [inline]
```

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

##### `_Bit_scan_forward()`

```
std::size_t __gnu_cxx::_Bit_scan_forward (
 std::size_t __num) [inline]
```

Generic Version of the `bsf` instruction.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object()`.

##### `iota()`

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

Create a range of sequentially increasing values.

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

**Parameters**

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

**operator"!="() [1/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

**operator"!="() [2/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test difference of string and C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

**operator"!="() [3/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of C string and string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

#### `operator+()` [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 _CharT __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate character and string.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

#### Returns

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::reserve\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::operator+\(\)](#).

#### `operator+()` [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 _CharT __rhs)
```

Concatenate string and character.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

#### Returns

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::reserve\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::operator+\(\)](#).

**operator+()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate two strings.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::size\(\)](#) and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::size\(\)](#).

**operator+()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs)
```

Concatenate string and C string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::size\(\)](#).

**operator+()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate C string and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::size\(\)](#).

**operator<()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string precedes string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string precedes C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).



**operator<()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string precedes string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<=()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't follow string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<=()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't follow C string.

**Parameters**

|                    |         |
|--------------------|---------|
| <code>__lhs</code> | String. |
|--------------------|---------|

|                    |           |
|--------------------|-----------|
| <code>__rhs</code> | C string. |
|--------------------|-----------|

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator<=()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't follow string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator==()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::data\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_All](#)

**operator==()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
bool __gnu_cxx::operator== (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test equivalence of string and C string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

#### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::data\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_All](#)

#### `operator==( )` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of C string and string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

#### `operator==( )` [4/4]

```
template<typename _Tp>
bool __gnu_cxx::operator== (
 const _Pointer_adapter< _Tp > & __lhs,
 const _Pointer_adapter< _Tp > & __rhs) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

#### `operator>( )` [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string follows string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string follows C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string follows string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>=()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't precede string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>=()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't precede C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

**operator>=()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't precede string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
|--------------------|-----------|

|                    |         |
|--------------------|---------|
| <code>__rhs</code> | String. |
|--------------------|---------|

#### Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::compare\(\)](#).

#### swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::swap (
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Swap contents of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base >::swap\(\)](#).

## 4.2 \_\_gnu\_cxx::\_\_detail Namespace Reference

### Classes

- class [\\_\\_mini\\_vector](#)
- class [\\_Bitmap\\_counter](#)
- class [\\_Ffit\\_finder](#)

### Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte` , `bits_per_block` }
- enum `_Tag` { `_S_leaf` , `_S_concat` , `_S_substringfn` , `_S_function` }

### Functions

- void [\\_\\_bit\\_allocate](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- void [\\_\\_bit\\_free](#) (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare>  
\_ForwardIterator [\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)

#### 4.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

### 4.2.2 Function Documentation

#### `__bit_allocate()`

```
void __gnu_cxx::__detail::__bit_allocate (
 std::size_t * __pmap,
 std::size_t __pos) throw () [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.  
Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1>::\\_\\_M\\_allocate\\_single\\_object\(\)](#).

#### `__bit_free()`

```
void __gnu_cxx::__detail::__bit_free (
 std::size_t * __pmap,
 std::size_t __pos) throw () [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.  
Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1>::\\_\\_M\\_deallocate\\_single\\_object\(\)](#).

#### `__num_bitmaps()`

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_bitmaps (
 _AddrPair __ap) [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.

References [\\_\\_num\\_blocks\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1>::\\_\\_M\\_allocate\\_single\\_object\(\)](#), and [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp1>::\\_\\_M\\_deallocate\\_single\\_object\(\)](#).

#### `__num_blocks()`

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_blocks (
 _AddrPair __ap) [inline]
```

The number of Blocks pointed to by the address pair passed to the function.

Referenced by [\\_\\_num\\_bitmaps\(\)](#).

## 4.3 `__gnu_cxx::typelist` Namespace Reference

### Functions

- `template<typename Fn, typename Typelist>`  
`void apply (Fn &, Typelist)`
- `template<typename Fn, typename Typelist>`  
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist>`  
`void apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>`  
`void apply_generator (Gn &, TypelistT, TypelistV)`

### 4.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

### 4.3.2 Function Documentation

#### **apply\_generator()**

```
template<typename Gn, typename Typelist>
void __gnu_cxx::typelist::apply_generator (
 Gn & ,
 Typelist)
```

Apply all typelist types to generator functor.

## 4.4 **\_\_gnu\_debug** Namespace Reference

### Classes

- class [\\_After\\_nth\\_from](#)
- struct [\\_BeforeBeginHelper](#)
- class [\\_Equal\\_to](#)
- class [\\_Not\\_equal\\_to](#)
- class [\\_Safe\\_container](#)
- class [\\_Safe\\_forward\\_list](#)
- class [\\_Safe\\_iterator](#)
- class [\\_Safe\\_iterator\\_base](#)
- class [\\_Safe\\_local\\_iterator](#)
- class [\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_Safe\\_node\\_sequence](#)
- class [\\_Safe\\_sequence](#)
- class [\\_Safe\\_sequence\\_base](#)
- class [\\_Safe\\_unordered\\_container](#)
- class [\\_Safe\\_unordered\\_container\\_base](#)
- class [\\_Safe\\_vector](#)
- struct [\\_Sequence\\_traits](#)
- class [basic\\_string](#)
- class [type\\_info](#)

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< char16\_t > **u16string**
- typedef [basic\\_string](#)< char32\_t > **u32string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

### Enumerations

- enum **\_Debug\_msg\_id** {  
    [\\_\\_msg\\_valid\\_range](#), [\\_\\_msg\\_insert\\_singular](#), [\\_\\_msg\\_insert\\_different](#), [\\_\\_msg\\_erase\\_bad](#),  
    [\\_\\_msg\\_erase\\_different](#), [\\_\\_msg\\_subscript\\_oob](#), [\\_\\_msg\\_empty](#), [\\_\\_msg\\_unpartitioned](#),  
    [\\_\\_msg\\_unpartitioned\\_pred](#), [\\_\\_msg\\_unsorted](#), [\\_\\_msg\\_unsorted\\_pred](#), [\\_\\_msg\\_not\\_heap](#),  
    [\\_\\_msg\\_not\\_heap\\_pred](#), [\\_\\_msg\\_bad\\_bitset\\_write](#), [\\_\\_msg\\_bad\\_bitset\\_read](#), [\\_\\_msg\\_bad\\_bitset\\_flip](#),  
    [\\_\\_msg\\_self\\_splice](#), [\\_\\_msg\\_splice\\_alloc](#), [\\_\\_msg\\_splice\\_bad](#), [\\_\\_msg\\_splice\\_other](#),  
    [\\_\\_msg\\_splice\\_overlap](#), [\\_\\_msg\\_init\\_singular](#), [\\_\\_msg\\_init\\_copy\\_singular](#), [\\_\\_msg\\_init\\_const\\_singular](#),  
    [\\_\\_msg\\_copy\\_singular](#), [\\_\\_msg\\_bad\\_deref](#), [\\_\\_msg\\_bad\\_inc](#), [\\_\\_msg\\_bad\\_dec](#),  
    [\\_\\_msg\\_iter\\_subscript\\_oob](#), [\\_\\_msg\\_advance\\_oob](#), [\\_\\_msg\\_retreat\\_oob](#), [\\_\\_msg\\_iter\\_compare\\_bad](#),  
    [\\_\\_msg\\_compare\\_different](#), [\\_\\_msg\\_iter\\_order\\_bad](#), [\\_\\_msg\\_order\\_different](#), [\\_\\_msg\\_distance\\_bad](#),  
    [\\_\\_msg\\_distance\\_different](#), [\\_\\_msg\\_deref\\_istream](#), [\\_\\_msg\\_inc\\_istream](#), [\\_\\_msg\\_output\\_ostream](#),  
}



```

__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end, __msg_erase_after_bad,
__msg_valid_range2, __msg_local_iter_compare_bad, __msg_non_empty_range, __msg_self_move↵
__assign,
__msg_bucket_index_oob, __msg_valid_load_factor, __msg_equal_allocs, __msg_insert_range↵
from_self,
__msg_irreflexive_ordering }
• enum __Distance_precision {
__dp_none, __dp_equality, __dp_sign, __dp_sign_max_size,
__dp_exact }

```

## Functions

- `template<typename _Iterator>`  
`constexpr _Iterator __base (_Iterator __it)`
- `template<typename _Iterator, typename _Sequence>`  
`constexpr _Iterator __base (const __Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &↵`  
`__it)`
- `template<typename _Iterator>`  
`constexpr auto __base (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__↵`  
`base(__it.base()))`
- `template<typename _Iterator, typename _Sequence>`  
`constexpr std::reverse_iterator< _Iterator > __base (const std::reverse_iterator< __Safe_iterator< _Iterator, ↵`  
`_Sequence, std::random_access_iterator_tag > > &__it)`
- `template<typename _InputIterator, typename _Size>`  
`constexpr bool __can_advance (_InputIterator, _Size)`
- `template<typename _InputIterator, typename _Diff>`  
`constexpr bool __can_advance (_InputIterator, const std::pair< _Diff, __Distance_precision > &, int)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>`  
`constexpr bool __can_advance (const __Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`constexpr bool __can_advance (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< ↵`  
`_Diff, __Distance_precision > &, int)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __can_advance (const std::move_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __can_advance (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff, ↵`  
`__Distance_precision > &__dist, int __way)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __can_advance (const std::reverse_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __can_advance (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff, ↵`  
`__Distance_precision > &__dist, int __way)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`value, _Pred __pred)`

- `template<typename _Iterator>`  
`constexpr bool __check_singular (_Iterator const &)`
- `template<typename _Tp>`  
`constexpr bool __check_singular (_Tp *const &__ptr)`
- `bool __check_singular_aux (const _Safe_iterator_base * __x)`
- `bool __check_singular_aux (const class _Safe_iterator_base *)`
- `bool __check_singular_aux (const void *)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _ForwardIterator>`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)`
- `template<typename _InputIterator>`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)`
- `template<typename _CharT, typename _Integer>`  
`const _CharT * __check_string (const _CharT * __s, _Integer __n, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT>`  
`const _CharT * __check_string (const _CharT * __s, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _InputIterator>`  
`constexpr _InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename _OtherCategory>`  
`bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &)`

- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux2 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const \_InputIterator &__other, const \_InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>`  
`bool \_\_foreign\_iterator\_aux2 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const \_Safe\_iterator< _OtherIterator, _Sequence, _Category > &__other, const \_Safe\_iterator< _OtherIterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux3 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const \_InputIterator &, const \_InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __foreign_iterator_aux3 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const \_InputIterator &__other, const \_InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __foreign_iterator_aux4 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &,...)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`bool __foreign_iterator_aux4 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_get\_distance ( _Iterator __lhs, _Iterator __rhs)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_get\_distance ( _Iterator __lhs, _Iterator __rhs, std::input\_iterator\_tag)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_get\_distance ( _Iterator __lhs, _Iterator __rhs, std::random\_access\_iterator\_tag)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_get\_distance (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr \_Distance\_traits< _Iterator >::__type \_\_get\_distance (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr bool \_\_is\_irreflexive ( _Iterator __it)`
- `template<typename _Iterator, typename _Pred>`  
`constexpr bool \_\_is\_irreflexive\_pred ( _Iterator __it, _Pred __pred)`
- `template<typename _Iterator>`  
`constexpr \_Iterator \_\_unsafe ( _Iterator __it)`
- `template<typename _Iterator, typename _Sequence>`  
`\_Iterator \_\_unsafe (const \_Safe\_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _UContainer>`  
`\_Iterator \_\_unsafe (const \_Safe\_local\_iterator< _Iterator, _UContainer > &__it)`
- `template<typename _Iterator>`  
`constexpr auto \_\_unsafe (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(\_\_unsafe(__it.base())))`
- `template<typename _Iterator>`  
`constexpr auto \_\_unsafe (const std::reverse\_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(\_\_unsafe(__it.base())))`
- `template<typename _InputIterator>`  
`constexpr bool \_\_valid\_range ( _InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`constexpr bool \_\_valid\_range ( _InputIterator __first, _InputIterator __last, typename \_Distance\_traits< \_InputIterator >::__type &__dist)`

- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const`  
`__Safe_iterator< _Iterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const`  
`__Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator< _`  
`Iterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator< _`  
`Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _UContainer > &__first, const __Safe_local_iterator<`  
`_Iterator, _UContainer > &__last)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __valid_range (const __Safe_local_iterator< _Iterator, _UContainer > &__first, const __Safe_local_iterator<`  
`_Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)`
- `template<typename _Iterator>`  
`constexpr bool __valid_range (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _`  
`Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator>`  
`constexpr bool __valid_range (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _`  
`_Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT, _Traits > &__is, basic_string<`  
`_CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT, _Traits > &__is, basic_string<`  
`_CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs, const basic_string< _CharT, _`  
`Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__`  
`lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__`  
`lhs, const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator== (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`void swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

#### 4.4.1 Detailed Description

GNU debug classes for public use.

#### 4.4.2 Typedef Documentation

##### u16string

```
typedef basic_string<char16_t> __gnu_debug::u16string
```

A string of `char16_t`.

##### u32string

```
typedef basic_string<char32_t> __gnu_debug::u32string
```

A string of `char32_t`.

#### 4.4.3 Enumeration Type Documentation

##### [\\_Distance\\_precision](#)

```
enum __gnu_debug::Distance_precision
```

The precision to which we can calculate the distance between two iterators.

#### 4.4.4 Function Documentation

##### [\\_\\_base\(\)](#)

```
template<typename _Iterator>
_Iterator __gnu_debug::__base (
 _Iterator __it) [inline], [constexpr]
```

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Referenced by [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::\\_\\_M\\_before\\_dereferenceable\(\)](#).

##### [\\_\\_check\\_singular\(\)](#)

```
template<typename _Tp>
bool __gnu_debug::__check_singular (
 _Tp *const & __ptr) [inline], [constexpr]
```

Non-NULL pointers are nonsingular.

##### [\\_\\_check\\_singular\\_aux\(\)](#)

```
bool __gnu_debug::__check_singular_aux (
 const _Safe_iterator_base * __x) [inline]
```

Iterators that derive from `_Safe_iterator_base` can be determined singular or non-singular.

References [\\_\\_gnu\\_debug::Safe\\_iterator\\_base::\\_\\_M\\_singular\(\)](#).

##### [\\_\\_check\\_string\(\)](#) [1/2]

```
template<typename _CharT, typename _Integer>
const _CharT * __gnu_debug::__check_string (
 const _CharT * __s,
 _Integer __n,
 const char * __file,
 unsigned int __line,
 const char * __function) [inline]
```

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

**`__check_string()`** [2/2]

```
template<typename _CharT>
const _CharT * __gnu_debug::__check_string (
 const _CharT * __s,
 const char * __file,
 unsigned int __line,
 const char * __function) [inline]
```

Checks that `__s` is non-NULL and then returns `__s`.

**`__foreign_iterator_aux2()`** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator,
typename _OtherSequence, typename _OtherCategory>
bool __gnu_debug::__foreign_iterator_aux2 (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & ,
 const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ,
 const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &) [inline]
```

Handle debug iterators from different types of container.

**`__foreign_iterator_aux2()`** [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
bool __gnu_debug::__foreign_iterator_aux2 (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __it,
 const _Safe_iterator< _OtherIterator, _Sequence, _Category > & __other,
 const _Safe_iterator< _OtherIterator, _Sequence, _Category > &) [inline]
```

Handle debug iterators from the same type of container.

**`__get_distance()`**

```
template<typename _Iterator>
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (
 _Iterator __lhs,
 _Iterator __rhs,
 std::random_access_iterator_tag) [inline], [constexpr]
```

Determine the distance between two iterators with some known precision.

References [std::make\\_pair\(\)](#).

**`__valid_range()`** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category>
bool __gnu_debug::__valid_range (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __first,
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __last,
 typename _Distance_traits< _Iterator >::__type & __dist) [inline], [constexpr]
```

Safe iterators know how to check if they form a valid range.

**`__valid_range()`** [2/2]

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::__valid_range (
 const _Safe_local_iterator< _Iterator, _UContainer > & __first,
 const _Safe_local_iterator< _Iterator, _UContainer > & __last,
 typename _Distance_traits< _Iterator >::__type & __dist_info) [inline]
```

Safe local iterators know how to check if they form a valid range.

## 4.5 `__gnu_internal` Namespace Reference

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 4.6 `__gnu_parallel` Namespace Reference

### Classes

- struct [\\_\\_accumulate\\_binop\\_reduct](#)
- struct [\\_\\_accumulate\\_selector](#)
- struct [\\_\\_adjacent\\_difference\\_selector](#)
- struct [\\_\\_adjacent\\_find\\_selector](#)
- class [\\_\\_binder1st](#)
- class [\\_\\_binder2nd](#)
- struct [\\_\\_count\\_if\\_selector](#)
- struct [\\_\\_count\\_selector](#)
- struct [\\_\\_fill\\_selector](#)
- struct [\\_\\_find\\_first\\_of\\_selector](#)
- struct [\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_for\\_each\\_selector](#)
- struct [\\_\\_generate\\_selector](#)
- struct [\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_generic\\_for\\_each\\_selector](#)
- struct [\\_\\_identity\\_selector](#)
- struct [\\_\\_inner\\_product\\_selector](#)
- struct [\\_\\_max\\_element\\_reduct](#)
- struct [\\_\\_min\\_element\\_reduct](#)
- struct [\\_\\_mismatch\\_selector](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterlIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAIterlIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterlIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_replace\\_if\\_selector](#)
- struct [\\_\\_replace\\_selector](#)
- struct [\\_\\_transform1\\_selector](#)
- struct [\\_\\_transform2\\_selector](#)
- class [\\_\\_unary\\_negate](#)
- struct [\\_DRandomShufflingGlobalData](#)
- struct [\\_DRSSorterPU](#)
- struct [\\_DummyReduct](#)
- class [\\_EqualFromLess](#)
- struct [\\_EqualTo](#)
- class [\\_GuardedIterator](#)
- class [\\_IteratorPair](#)
- class [\\_IteratorTriple](#)
- struct [\\_Job](#)
- struct [\\_Less](#)



- class [\\_Lexicographic](#)
- class [\\_LexicographicReverse](#)
- class [\\_LoserTree](#)
- class [\\_LoserTree< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeBase](#)
- class [\\_LoserTreePointer](#)
- class [\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerBase](#)
- class [\\_LoserTreePointerUnguarded](#)
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerUnguardedBase](#)
- struct [\\_LoserTreeTraits](#)
- class [\\_LoserTreeUnguarded](#)
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeUnguardedBase](#)
- struct [\\_Multiplies](#)
- struct [\\_Nothing](#)
- struct [\\_Piece](#)
- struct [\\_Plus](#)
- struct [\\_PMWMSSortingData](#)
- class [\\_PseudoSequence](#)
- class [\\_PseudoSequenceIterator](#)
- struct [\\_QSBThreadLocal](#)
- class [\\_RandomNumber](#)
- class [\\_RestrictedBoundedConcurrentQueue](#)
- struct [\\_SamplingSorter](#)
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)
- struct [\\_Settings](#)
- struct [\\_SplitConsistently](#)
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [balanced\\_quicksort\\_tag](#)
- struct [balanced\\_tag](#)
- struct [constant\\_size\\_blocks\\_tag](#)
- struct [default\\_parallel\\_tag](#)
- struct [equal\\_split\\_tag](#)
- struct [exact\\_tag](#)
- struct [find\\_tag](#)
- struct [growing\\_blocks\\_tag](#)
- struct [multiway\\_mergesort\\_exact\\_tag](#)
- struct [multiway\\_mergesort\\_sampling\\_tag](#)
- struct [multiway\\_mergesort\\_tag](#)
- struct [omp\\_loop\\_static\\_tag](#)
- struct [omp\\_loop\\_tag](#)
- struct [parallel\\_tag](#)
- struct [quicksort\\_tag](#)
- struct [sampling\\_tag](#)
- struct [sequential\\_tag](#)
- struct [unbalanced\\_tag](#)

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic** , **force\_sequential** , **force\_parallel** }
- enum [\\_FindAlgorithm](#) { **GROWING\_BLOCKS** , **CONSTANT\_SIZE\_BLOCKS** , **EQUAL\_SPLIT** }
- enum [\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_Parallelism](#) { **sequential** , **parallel\_unbalanced** , **parallel\_balanced** , **parallel\_omp\_loop** , **parallel\_omp\_loop\_static** , **parallel\_taskqueue** }
- enum [\\_PartialSumAlgorithm](#) { **RECURSIVE** , **LINEAR** }
- enum [\\_SortAlgorithm](#) { **MWMS** , **QS** , **QS\_BALANCED** }
- enum [\\_SplittingAlgorithm](#) { **SAMPLING** , **EXACT** }

## Functions

- template<typename \_Tp>  
\_Tp [\\_\\_add\\_omp](#) (volatile \_Tp \* \_\_ptr, \_Tp \_\_addend)
- template<typename \_RAIter, typename \_DifferenceTp>  
void [\\_\\_calc\\_borders](#) (\_RAIter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \* \_\_off)
- template<typename \_Tp>  
bool [\\_\\_cas\\_omp](#) (volatile \_Tp \* \_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp>  
bool [\\_\\_compare\\_and\\_swap](#) (volatile \_Tp \* \_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Iter, typename \_OutputIterator>  
\_OutputIterator [\\_\\_copy\\_tail](#) (std::pair< \_Iter, \_Iter > \_\_b, std::pair< \_Iter, \_Iter > \_\_e, \_OutputIterator \_\_r)
- void [\\_\\_decode2](#) (\_CASable \_\_x, int & \_\_a, int & \_\_b)
- template<typename \_RAIter, typename \_DifferenceTp>  
void [\\_\\_determine\\_samples](#) (\_PMWMSortingData< \_RAIter > \* \_\_sd, \_DifferenceTp \_\_num\_samples)
- [\\_CASable](#) [\\_\\_encode2](#) (int \_\_a, int \_\_b)
- template<typename \_DifferenceType, typename \_OutputIterator>  
\_OutputIterator [\\_\\_equally\\_split](#) (\_DifferenceType \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType>  
\_DifferenceType [\\_\\_equally\\_split\\_point](#) (\_DifferenceType \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, [\\_ThreadIndex](#) \_\_thread\_no)
- template<typename \_Tp>  
\_Tp [\\_\\_fetch\\_and\\_add](#) (volatile \_Tp \* \_\_ptr, \_Tp \_\_addend)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector>  
std::pair< \_RAIter1, \_RAIter2 > [\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector)
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector>  
std::pair< \_RAIter1, \_RAIter2 > [\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [constant\\_size\\_blocks\\_tag](#))
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector>  
std::pair< \_RAIter1, \_RAIter2 > [\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [equal\\_split\\_tag](#))
- template<typename \_RAIter1, typename \_RAIter2, typename \_Pred, typename \_Selector>  
std::pair< \_RAIter1, \_RAIter2 > [\\_\\_find\\_template](#) (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [growing\\_blocks\\_tag](#))

- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>`  
`_UserOp __for_each_template_random_access` (`_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality`  
`& __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator\_traits<`  
`_Iter >::difference_type __bound, \_Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_ed` (`_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,`  
`_Result __base, _Result & __output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_omp_loop` (`_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f,  $\leftrightarrow$`   
`Red __r, _Result __base, _Result & __output, typename std::iterator\_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_omp_loop_static` (`_RAIter __begin, _RAIter __end, _Op __o, _Fu`  
`& __f, _Red __r, _Result __base, _Result & __output, typename std::iterator\_traits< _RAIter >::difference_type`  
`__bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op __for_each_template_random_access_workstealing` (`_RAIter __begin, _RAIter __end, _Op __op, _Fu &  $\leftrightarrow$`   
`__f, _Red __r, _Result __base, _Result & __output, typename std::iterator\_traits< _RAIter >::difference_type`  
`__bound)`
- `\_ThreadIndex __get_max_threads` ()
- `bool __is_parallel` (const `\_Parallelism __p`)
- `template<typename _Iter, typename _Compare>`  
`bool __is_sorted` (`_Iter __begin, _Iter __end, _Compare __comp`)
- `template<typename _RAIter, typename _Compare>`  
`_RAIter __median_of_three_iterators` (`_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp`)
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance` (`_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  $\leftrightarrow$`   
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp`)
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance_movc` (`_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2`  
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp`)
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator __merge_advance_usual` (`_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2`  
`__end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp`)
- `template<typename _RAIter1, typename _RAIter3, typename _Compare>`  
`_RAIter3 __parallel_merge_advance` (`_RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1  $\leftrightarrow$`   
`__end2, _RAIter3 __target, typename std::iterator\_traits< _RAIter1 >::difference_type __max_length, _Compare`  
`__comp`)
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>`  
`_RAIter3 __parallel_merge_advance` (`_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  $\leftrightarrow$`   
`__end2, _RAIter3 __target, typename std::iterator\_traits< _RAIter1 >::difference_type __max_length, _Compare`  
`__comp`)
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_nth_element` (`_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp`)
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_partial_sort` (`_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp`)
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum` (`_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation`  
`__bin_op`)
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum_basecase` (`_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$`   
`_BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __parallel_partial_sum_linear` (`_Iter __begin, _Iter __end, _OutputIterator __result, _Binary $\leftrightarrow$`   
`Operation __bin_op, typename std::iterator\_traits< _Iter >::difference_type __n)`

- `template<typename _RAIter, typename _Predicate>`  
`std::iterator_traits< _RAIter >::difference_type __parallel_partition ( _RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng= _RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle_drs ( _RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __parallel_random_shuffle_drs_pu ( _DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)`
- `template<typename _IIter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_difference ( _IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_intersection ( _IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter, typename _OutputIterator, typename _Operation>`  
`_OutputIterator __parallel_set_operation ( _IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _IIter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_symmetric_difference ( _IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _IIter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __parallel_set_union ( _IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __parallel_sort ( _RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qs ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qs_conquer ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide ( _RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

- `template<typename _RAIter, typename _Compare>`  
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator>`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate>`  
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter, typename _Compare>`  
`void __qsb_conquer (_QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __iam, \_ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator\_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > ** __tls, _Compare & __comp, \_ThreadIndex __iam, bool __wait)`
- `template<typename _RandomNumberGenerator>`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator & __rng)`
- `template<typename _Size>`  
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp>`  
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred>`  
`__RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type>::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator & __rng)`
- `template<typename _Iter>`  
`void __shrink (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length)`
- `template<typename _Iter>`  
`void __shrink_and_double (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice)`
- `void __yield ()`
- `template<typename _Iter, typename _FuncType>`  
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FuncType & __f, int __oversampling=0)`
- `template<typename _Tp>`  
`const _Tp & max (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp>`  
`const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator\_traits< typename std::iterator\_traits< _RanSeqs >::value_type>::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType & __offset, _Compare __comp=std::less< _Tp >())`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <-`  
`__target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <-`  
`__target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <-`  
`__target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <-`  
`__target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut <-`  
`__target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename <-`  
`DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_3_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <-`  
`target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename <-`  
`DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_4_variant ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <-`  
`target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>`  
`void multiway_merge_exact_splitting ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _Difference<-`  
`Type __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, <-`  
`DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 <-`  
`__target, _DifferenceTp __length, _Compare __comp)`
- `template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree_sentinel ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, <-`  
`RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value<-`  
`type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`  
`_RAIter3 multiway_merge_loser_tree_unguarded ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, <-`  
`_RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value<-`  
`_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>`  
`void multiway_merge_sampling_splitting ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, <-`  
`DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< <-`  
`DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <-`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <-`  
`RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <-`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, <-`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, type-`  
`name _Compare>`  
`_RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↵`  
`target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare>`  
`void parallel_sort_mwms_pu (\_PMWMSortingData<_RAIter> * __sd, _Compare & __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.



## 4.6.2 Typedef Documentation

### **\_BinIndex**

```
typedef unsigned short __gnu_parallel::_BinIndex
```

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

### **\_CASable**

```
typedef int64_t __gnu_parallel::_CASable
```

Longest compare-and-swappable integer type on this platform.

### **\_SequenceIndex**

```
typedef uint64_t __gnu_parallel::_SequenceIndex
```

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

### **\_ThreadIndex**

```
typedef uint16_t __gnu_parallel::_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

## 4.6.3 Enumeration Type Documentation

### **\_AlgorithmStrategy**

```
enum __gnu_parallel::_AlgorithmStrategy
```

Strategies for run-time algorithm selection:

### **\_FindAlgorithm**

```
enum __gnu_parallel::_FindAlgorithm
```

Find algorithms:

### **\_MultiwayMergeAlgorithm**

```
enum __gnu_parallel::_MultiwayMergeAlgorithm
```

Merging algorithms:

### **\_Parallelism**

```
enum __gnu_parallel::_Parallelism
```

Run-time equivalents for the compile-time tags.

### Enumerator

|                                       |                                              |
|---------------------------------------|----------------------------------------------|
| <code>sequential</code>               | Not parallel.                                |
| <code>parallel_unbalanced</code>      | Parallel unbalanced (equal-sized chunks).    |
| <code>parallel_balanced</code>        | Parallel balanced (work-stealing).           |
| <code>parallel_omp_loop</code>        | Parallel with OpenMP dynamic load-balancing. |
| <code>parallel_omp_loop_static</code> | Parallel with OpenMP static load-balancing.  |
| <code>parallel_taskqueue</code>       | Parallel with OpenMP taskqueue construct.    |



**`_PartialSumAlgorithm`**

enum `__gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

**`_SortAlgorithm`**

enum `__gnu_parallel::_SortAlgorithm`

Sorting algorithms:

**`_SplittingAlgorithm`**

enum `__gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: sampling, `__exact`.

**4.6.4 Function Documentation****`__calc_borders()`**

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__calc_borders (
 _RAIter __elements,
 _DifferenceTp __length,
 _DifferenceTp * __off)
```

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

**Parameters**

|                         |                                           |
|-------------------------|-------------------------------------------|
| <code>__elements</code> | Begin iterator of sequence to search for. |
| <code>__length</code>   | Length of sequence to search for.         |
| <code>__off</code>      | Returned <code>__offsets</code> .         |

Referenced by `__search_template()`.

**`__compare_and_swap()`**

```
template<typename _Tp>
bool __gnu_parallel::__compare_and_swap (
 volatile _Tp * __ptr,
 _Tp __comparand,
 _Tp __replacement) [inline]
```

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__ptr</code>         | Pointer to signed integer. |
| <code>__comparand</code>   | Compare value.             |
| <code>__replacement</code> | Replacement value.         |

Referenced by `__parallel_partition()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>::pop_back()`, and `__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp>::pop_front()`.

**\_\_decode2()**

```
void __gnu_parallel::__decode2 (
 _CASable __x,
 int & __a,
 int & __b) [inline]
```

Decode two integers from one `gnu_parallel::_CASable`.

**Parameters**

|                  |                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------|
| <code>__x</code> | <code>__gnu_parallel::_CASable</code> to decode integers from.                                                 |
| <code>__a</code> | First integer, to be decoded from the most-significant <code>_CASable_bits/2</code> bits of <code>__x</code> . |
| <code>__b</code> | Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits of <code>__x</code> . |

**See also**

`__encode2`

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`.

**\_\_determine\_samples()**

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__determine_samples (
 _PMWMSortingData<_RAIter> * __sd,
 _DifferenceTp __num_samples)
```

Select `_M_samples` from a sequence.

**Parameters**

|                            |                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------|
| <code>__sd</code>          | Pointer to algorithm data. Result will be placed in <code>__sd-&gt;_M_samples</code> . |
| <code>__num_samples</code> | Number of <code>_M_samples</code> to select.                                           |

References `__equally_split()`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_samples`, `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_starts`, and `__gnu_parallel::_PMWMSortingData<_RAIter>::_M_starts`.

**\_\_encode2()**

```
_CASable __gnu_parallel::__encode2 (
 int __a,
 int __b) [inline]
```

Encode two integers into one `gnu_parallel::_CASable`.

**Parameters**

|                  |                                                                                         |
|------------------|-----------------------------------------------------------------------------------------|
| <code>__a</code> | First integer, to be encoded in the most-significant <code>_CASable_bits/2</code> bits. |
|------------------|-----------------------------------------------------------------------------------------|

|                  |                                                                                           |
|------------------|-------------------------------------------------------------------------------------------|
| <code>__b</code> | Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits. |
|------------------|-------------------------------------------------------------------------------------------|

**Returns**

value encoding `__a` and `__b`.

**See also**

`__decode2`

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`.

**`__equally_split()`**

```
template<typename _DifferenceType, typename _OutputIterator>
_OutputIterator __gnu_parallel::__equally_split (
 _DifferenceType __n,
 _ThreadIndex __num_threads,
 _OutputIterator __s)
```

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0, __n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

**Parameters**

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__s</code>           | Splitters          |

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**`__equally_split_point()`**

```
template<typename _DifferenceType>
_DifferenceType __gnu_parallel::__equally_split_point (
 _DifferenceType __n,
 _ThreadIndex __num_threads,
 _ThreadIndex __thread_no)
```

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

**Parameters**

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__thread_no</code>   | Number of threads  |

**Returns**

splitting point

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#).

**`__fetch_and_add()`**

```
template<typename _Tp>
_Tp __gnu_parallel::__fetch_and_add (
 volatile _Tp * __ptr,
 _Tp __addend) [inline]
```

Add a value to a variable, atomically.

**Parameters**

|                       |                              |
|-----------------------|------------------------------|
| <code>__ptr</code>    | Pointer to a signed integer. |
| <code>__addend</code> | Value to add.                |

Referenced by [\\_\\_find\\_template\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), [\\_\\_parallel\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>::push\\_front\(\)](#).

**`__find_template()` [1/4]**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair<_RAIter1, _RAIter2 > __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector) [inline]
```

Parallel `std::find`, switch for different algorithms.

**Parameters**

|                         |                                                                                     |
|-------------------------|-------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                   |
| <code>__end1</code>     | End iterator of first sequence.                                                     |
| <code>__begin2</code>   | Begin iterator of second sequence. Must have same length as first sequence.         |
| <code>__pred</code>     | Find predicate.                                                                     |
| <code>__selector</code> | _Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...) |

**Returns**

Place of finding in both sequences.

References [\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [std::make\\_pair\(\)](#).

Referenced by [\\_\\_find\\_template\(\)](#).

**`__find_template()` [2/4]**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 constant_size_blocks_tag)
```

Parallel `std::find`, constant block size variant.

**Parameters**

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

**Returns**

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_initial\\_block\\_size](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#), [std::pair<\\_T1, \\_T2>::first](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [std::min\(\)](#).

**`__find_template()` [3/4]**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 equal_split_tag)
```

Parallel `std::find`, equal splitting variant.

#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

#### Returns

Place of finding in both sequences.

References [\\_\\_equally\\_split\(\)](#), and [\\_GLIBCXX\\_CALL](#).

#### `__find_template()` [4/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 growing_blocks_tag)
```

Parallel `std::find`, growing block size variant.

#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

#### Returns

Place of finding in both sequences.

#### See also

`__gnu_parallel::_Settings::find_sequential_search_size`

`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References [\\_\\_fetch\\_and\\_add\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_scale\\_factor](#), [\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#), [std::pair<\\_T1, \\_T2>::first](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), and [std::min\(\)](#).

**`__for_each_template_random_access()`**

```
template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
_Result>
__UserOp __gnu_parallel::__for_each_template_random_access (
 _IIter __begin,
 _IIter __end,
 _UserOp __user_op,
 _Functionality & __functionality,
 _Red __reduction,
 _Result __reduction_start,
 _Result & __output,
 typename std::iterator_traits< _IIter >::difference_type __bound,
 _Parallelism __parallelism_tag)
```

Chose the desired algorithm by evaluating `__parallelism_tag`.

**Parameters**

|                                |                                                                                                                                                   |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>           | Begin iterator of input sequence.                                                                                                                 |
| <code>__end</code>             | End iterator of input sequence.                                                                                                                   |
| <code>__user_op</code>         | A user-specified functor (comparator, predicate, associative operator,...)                                                                        |
| <code>__functionality</code>   | functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. accumulate, <code>for_each</code> ,...) |
| <code>__reduction</code>       | Reduction functor.                                                                                                                                |
| <code>__reduction_start</code> | Initial value for reduction.                                                                                                                      |
| <code>__output</code>          | Output iterator.                                                                                                                                  |
| <code>__bound</code>           | Maximum number of elements processed.                                                                                                             |
| <code>__parallelism_tag</code> | Parallelization method                                                                                                                            |

References [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static\(\)](#), [parallel\\_omp\\_loop](#), [parallel\\_omp\\_loop\\_static](#), and [parallel\\_unbalanced](#).

**`__for_each_template_random_access_ed()`**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Result>
__Op __gnu_parallel::__for_each_template_random_access_ed (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__begin</code> | Begin iterator of element sequence. |
| <code>__end</code>   | End iterator of element sequence.   |

|                       |                                                                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...)                                                                       |
| <code>__f</code>      | Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                |
| <code>__output</code> | Pointer to position where final result is written to                                                                                     |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                           |

#### Returns

User-supplied functor (that may contain a part of the result).

References [\\_\\_equally\\_split\\_point\(\)](#), and [min\(\)](#).

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

#### `__for_each_template_random_access_omp_loop()`

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

#### Parameters

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code> | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>   | End iterator of element sequence.                                                                                                             |
| <code>__o</code>     | User-supplied functor (comparator, predicate, adding functor, etc.).                                                                          |
| <code>__f</code>     | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |



|                       |                                                                                                                    |
|-----------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality). |
| <code>__base</code>   | Base value for reduction.                                                                                          |
| <code>__output</code> | Pointer to position where final result is written to                                                               |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                     |

**Returns**

User-supplied functor (that may contain a part of the result).

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

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

**`__for_each_template_random_access_omp_loop_static()`**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)

```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

**Parameters**

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).             |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

**Returns**

User-supplied functor (that may contain a part of the result).

References [std::min\(\)](#).

**\_\_for\_each\_template\_random\_access\_workstealing()**

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_workstealing (
 _RAIter __begin,
 _RAIter __end,
 _Op __op,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

**Parameters**

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__op</code>     | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

**Returns**

User-supplied functor (that may contain a part of the result).

References [\\_\\_fetch\\_and\\_add\(\)](#), [\\_\\_yield\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::Job<\\_DifferenceTp>::\\_\\_M\\_first](#), [\\_\\_gnu\\_parallel::Job<\\_DifferenceTp>::\\_\\_M\\_last](#), [\\_\\_gnu\\_parallel::Job<\\_DifferenceTp>::\\_\\_M\\_load](#), [\\_\\_gnu\\_parallel::Settings::cache\\_l](#), [\\_\\_gnu\\_parallel::Settings::get\(\)](#), [max\(\)](#), and [min\(\)](#).

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\(\)](#).

**\_\_is\_sorted()**

```
template<typename _IIter, typename _Compare>
bool __gnu_parallel::__is_sorted (
```

```

 __Iter __begin,
 __Iter __end,
 __Compare __comp)

```

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

#### Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__begin</code> | Begin iterator of sequence. |
| <code>__end</code>   | End iterator of sequence.   |
| <code>__comp</code>  | Comparator.                 |

#### Returns

`true` if sorted, `false` otherwise.

Referenced by [\\_\\_sequential\\_multiway\\_merge\(\)](#), [multiway\\_merge\\_loser\\_tree\\_sentinel\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

#### `__median_of_three_iterators()`

```

template<typename _RAIter, typename _Compare>
_RAIter __gnu_parallel::__median_of_three_iterators (
 _RAIter __a,
 _RAIter __b,
 _RAIter __c,
 _Compare __comp)

```

Compute the median of three referenced elements, according to `__comp`.

#### Parameters

|                     |                  |
|---------------------|------------------|
| <code>__a</code>    | First iterator.  |
| <code>__b</code>    | Second iterator. |
| <code>__c</code>    | Third iterator.  |
| <code>__comp</code> | Comparator.      |

Referenced by [\\_\\_qsb\\_divide\(\)](#).

#### `__merge_advance()`

```

template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp) [inline]

```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

#### Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

#### Returns

Output end iterator.

References [\\_\\_merge\\_advance\\_movc\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#), and [\\_\\_sequential\\_multiway\\_merge\(\)](#).

#### `__merge_advance_movc()`

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_movc (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

#### Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

Referenced by [\\_\\_merge\\_advance\(\)](#).

**`__merge_advance_usual()`**

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_usual (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

**`__parallel_merge_advance()` [1/2]**

```
template<typename _RAIter1, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter1 & __begin2,
 _RAIter1 __end2,
 _RAIter3 __target,
 typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
 _Compare __comp) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

References [std::make\\_pair\(\)](#), [multiway\\_merge\\_exact\\_splitting\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

**`__parallel_merge_advance()` [2/2]**

```
template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _RAIter3 __target,
 typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
 _Compare __comp) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

References [\\_\\_merge\\_advance\(\)](#).

**`__parallel_nth_element()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_nth_element (
 _RAIter __begin,
```

```

 _RAIter __nth,
 _RAIter __end,
 _Compare __comp)

```

Parallel implementation of `std::nth_element()`.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__begin</code> | Begin iterator of input sequence.                         |
| <code>__nth</code>   | _Iterator of element that must be in position afterwards. |
| <code>__end</code>   | End iterator of input sequence.                           |
| <code>__comp</code>  | Comparator.                                               |

References [\\_\\_parallel\\_partition\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::nth\\_element](#) and [\\_\\_gnu\\_parallel::\\_Settings::partition\\_minimal\\_n](#).

Referenced by [\\_\\_parallel\\_partial\\_sort\(\)](#).

#### `__parallel_partial_sort()`

```

template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_partial_sort (
 _RAIter __begin,
 _RAIter __middle,
 _RAIter __end,
 _Compare __comp)

```

Parallel implementation of `std::partial_sort()`.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence. |
| <code>__middle</code> | Sort until this position.         |
| <code>__end</code>    | End iterator of input sequence.   |
| <code>__comp</code>   | Comparator.                       |

References [\\_\\_parallel\\_nth\\_element\(\)](#).

#### `__parallel_partial_sum()`

```

template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op)

```

Parallel partial sum front-`__end`.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.  |
| <code>__end</code>    | End iterator of input sequence.    |
| <code>__result</code> | Begin iterator of output sequence. |

|                       |                              |
|-----------------------|------------------------------|
| <code>__bin_op</code> | Associative binary function. |
|-----------------------|------------------------------|

**Returns**

End iterator of output sequence.

References [\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), [\\_GLIBCXX\\_CALL](#), and [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#).

**`__parallel_partial_sum_basecase()`**

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op,
 typename std::iterator_traits< _IIter >::value_type __value)
```

Base case prefix sum routine.

**Parameters**

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.                                            |
| <code>__end</code>    | End iterator of input sequence.                                              |
| <code>__result</code> | Begin iterator of output sequence.                                           |
| <code>__bin_op</code> | Associative binary function.                                                 |
| <code>__value</code>  | Start value. Must be passed since the neutral element is unknown in general. |

**Returns**

End iterator of output sequence.

Referenced by [\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#).

**`__parallel_partial_sum_linear()`**

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op,
 typename std::iterator_traits< _IIter >::difference_type __n)
```

Parallel partial sum implementation, two-phase approach, no recursion.

**Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.  |
| <code>__end</code>    | End iterator of input sequence.    |
| <code>__result</code> | Begin iterator of output sequence. |
| <code>__bin_op</code> | Associative binary function.       |



|                  |                     |
|------------------|---------------------|
| <code>__n</code> | Length of sequence. |
|------------------|---------------------|

**Returns**

End iterator of output sequence.

References [\\_\\_equally\\_split\(\)](#), [\\_\\_parallel\\_partial\\_sum\\_basecase\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [std::min\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::partial\\_sum\\_dilation](#).

Referenced by [\\_\\_parallel\\_partial\\_sum\(\)](#).

**`__parallel_partition()`**

```
template<typename _RAIter, typename _Predicate>
std::iterator_traits<_RAIter >::difference_type __gnu_parallel::__parallel_partition (
 _RAIter __begin,
 _RAIter __end,
 _Predicate __pred,
 _ThreadIndex __num_threads)
```

Parallel implementation of `std::partition`.

**Parameters**

|                            |                                                             |
|----------------------------|-------------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence to split.                  |
| <code>__end</code>         | End iterator of input sequence to split.                    |
| <code>__pred</code>        | Partition predicate, possibly including some kind of pivot. |
| <code>__num_threads</code> | Maximum number of threads to use for this task.             |

**Returns**

Number of elements not fulfilling the predicate.

References [\\_\\_compare\\_and\\_swap\(\)](#), [\\_\\_fetch\\_and\\_add\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_VOLATILE](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::max\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::partition\\_chunk\\_share](#), and [\\_\\_gnu\\_parallel::\\_Settings::partition\\_chunk\\_size](#).

Referenced by [\\_\\_parallel\\_nth\\_element\(\)](#), [\\_\\_parallel\\_sort\\_qs\\_divide\(\)](#), and [\\_\\_qsb\\_divide\(\)](#).

**`__parallel_random_shuffle()`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle (
 _RAIter __begin,
 _RAIter __end,
 _RandomNumberGenerator __rng = _RandomNumber()) [inline]
```

Parallel random public call.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

References [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**\_\_parallel\_random\_shuffle\_drs()**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs (
 _RAIter __begin,
 _RAIter __end,
 typename std::iterator_traits< _RAIter >::difference_type __n,
 _ThreadIndex __num_threads,
 _RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

**Parameters**

|                            |                                 |
|----------------------------|---------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.     |
| <code>__end</code>         | End iterator of sequence.       |
| <code>__n</code>           | Length of sequence.             |
| <code>__num_threads</code> | Number of threads to use.       |
| <code>__rng</code>         | Random number generator to use. |

References [\\_\\_gnu\\_parallel::\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_bins\\_end](#), [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_bins\\_begin](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_num\\_bins](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_M\\_seed](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_temporaries](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::L2\\_cache\\_size](#), [std::max\(\)](#), [std::numeric\\_limits< \\_Tp >::max\(\)](#), [std::min\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::TLB](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\(\)](#).

**\_\_parallel\_random\_shuffle\_drs\_pu()**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
 _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus)
```

Random shuffle code executed by each thread.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pus</code> | Array of thread-local data records. |
|--------------------|-------------------------------------|

References [\\_\\_random\\_number\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_bins\\_begin](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_num\\_bins](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >::\\_M\\_seed](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_starts](#), [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >::\\_M\\_temporaries](#), and [std::partial\\_sum\(\)](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#).

**\_\_parallel\_sort()** [1/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
```

```
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 balanced_quicksort_tag __parallelism) [inline]
```

Choose balanced quicksort for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `_GLIBCXX_CALL`.  
Here is the call graph for this function:



#### `__parallel_sort()` [2/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 default_parallel_tag __parallelism) [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Here is the call graph for this function:



### `__parallel_sort()` [3/7]

```

template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_exact_tag __parallelism) [inline]

```

Choose multiway mergesort with exact splitting, for parallel sorting.

### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_GLIBCXX\\_CALL](#), and [parallel\\_sort\\_mwms\(\)](#).

Here is the call graph for this function:



**`__parallel_sort()`** [4/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_sampling_tag __parallelism) [inline]
```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

**Template Parameters**

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_GLIBCXX\\_CALL](#), and [parallel\\_sort\\_mwms\(\)](#).

Here is the call graph for this function:

**`__parallel_sort()`** [5/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_tag __parallelism) [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [parallel\\_sort\\_mwms\(\)](#).  
Here is the call graph for this function:



`__parallel_sort()` [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::_parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 parallel_tag __parallelism) [inline]
```

Choose a parallel sorting algorithm.

Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

References [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads\(\)](#), [\\_\\_parallel\\_sort\\_qs\(\)](#), [\\_\\_parallel\\_sort\\_qsb\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [parallel\\_sort\\_mwms\(\)](#).

[illegible]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 quicksort_tag __parallelism) [inline]
```

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

|                         |              |
|-------------------------|--------------|
| <u>  </u> <i>stable</i> | Sort stable. |
|-------------------------|--------------|

Here is the call graph for this function:



**\_\_parallel\_sort\_qs()**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort main call.

**Parameters**

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence.                        |
| <code>__end</code>         | End iterator input sequence, ignored.                    |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

References [\\_\\_parallel\\_sort\\_qs\\_conquer\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Referenced by [\\_\\_parallel\\_sort\(\)](#), and [\\_\\_parallel\\_sort\(\)](#).

**\_\_parallel\_sort\_qs\_conquer()**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs_conquer (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort conquer step.

**Parameters**

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

References [\\_\\_parallel\\_sort\\_qs\\_conquer\(\)](#), [\\_\\_parallel\\_sort\\_qs\\_divide\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#).

Referenced by [\\_\\_parallel\\_sort\\_qs\(\)](#), and [\\_\\_parallel\\_sort\\_qs\\_conquer\(\)](#).

**\_\_parallel\_sort\_qs\_divide()**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 typename std::iterator_traits<_RAIter >::difference_type __pivot_rank,
 typename std::iterator_traits<_RAIter >::difference_type __num_samples,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.



**Parameters**

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__pivot_rank</code>  | Desired <code>__rank</code> of the pivot.                |
| <code>__num_samples</code> | Choose pivot from that many samples.                     |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

References [\\_\\_parallel\\_partition\(\)](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_sort\\_qs\\_conquer\(\)](#).

**`__parallel_sort_qsb()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qsb (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Top-level quicksort routine.

**Parameters**

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.                              |
| <code>__end</code>         | End iterator of sequence.                                |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

References [\\_\\_qsb\\_conquer\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_GLIBCXX\\_CALL](#), and [std::make\\_pair\(\)](#).

Referenced by [\\_\\_parallel\\_sort\(\)](#), and [\\_\\_parallel\\_sort\(\)](#).

**`__parallel_unique_copy()` [1/2]**

```
template<typename _IIter, class _OutputIterator>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
 _IIter __first,
 _IIter __last,
 _OutputIterator __result) [inline]
```

Parallel `std::unique_copy()`, without explicit equality predicate.

**Parameters**

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <code>__first</code>  | Begin iterator of input sequence.                  |
| <code>__last</code>   | End iterator of input sequence.                    |
| <code>__result</code> | Begin iterator of result <code>__sequence</code> . |

**Returns**

End iterator of result `__sequence`.

References [\\_\\_parallel\\_unique\\_copy\(\)](#).

**`__parallel_unique_copy()` [2/2]**

```
template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
 _IIter __first,
 _IIter __last,
 _OutputIterator __result,
 _BinaryPredicate __binary_pred)
```

Parallel `std::unique_copy()`, w/`__o` explicit equality predicate.

**Parameters**

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>__first</code>       | Begin iterator of input sequence.                  |
| <code>__last</code>        | End iterator of input sequence.                    |
| <code>__result</code>      | Begin iterator of result <code>__sequence</code> . |
| <code>__binary_pred</code> | Equality predicate.                                |

**Returns**

End iterator of result `__sequence`.

References [\\_\\_equally\\_split\(\)](#), and [\\_GLIBCXX\\_CALL](#).

Referenced by [\\_\\_parallel\\_unique\\_copy\(\)](#).

**`__qsb_conquer()`**

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_conquer (
 _QSBThreadLocal< _RAIter > ** __tls,
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __iam,
 _ThreadIndex __num_threads,
 bool __parent_wait)
```

Quicksort conquer step.

**Parameters**

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__tls</code>         | Array of thread-local storages.                          |
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__iam</code>         | Number of the thread processing this function.           |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

References [\\_\\_qsb\\_conquer\(\)](#), [\\_\\_qsb\\_divide\(\)](#), [\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_elements\\_leftover](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_initial](#), [std::max\(\)](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_sort\\_qsb\(\)](#), and [\\_\\_qsb\\_conquer\(\)](#).

### `__qsb_divide()`

```
template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Balanced quicksort divide step.

#### Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

#### Precondition

`(__end-__begin)>=1`

References [\\_\\_median\\_of\\_three\\_iterators\(\)](#), and [\\_\\_parallel\\_partition\(\)](#).

Referenced by [\\_\\_qsb\\_conquer\(\)](#).

### `__qsb_local_sort_with_helping()`

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_local_sort_with_helping (
 _QSBThreadLocal<_RAIter> ** __tls,
 _Compare & __comp,
 _ThreadIndex __iam,
 bool __wait)
```

Quicksort step doing load-balanced local sort.

#### Parameters

|                     |                                                |
|---------------------|------------------------------------------------|
| <code>__tls</code>  | Array of thread-local storages.                |
| <code>__comp</code> | Comparator.                                    |
| <code>__iam</code>  | Number of the thread processing this function. |

References [\\_\\_yield\(\)](#), [GLIBCXX\\_PARALLEL\\_ASSERTIONS](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_elements\\_leftover](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_initial](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_leftover\\_parts](#), [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>::\\_\\_M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::\\_\\_Settings::get\(\)](#), [std::make\\_pair\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_Settings::sort\\_qsb\\_base\\_case\\_maximal\\_n](#).

Referenced by [\\_\\_qsb\\_conquer\(\)](#).

### `__random_number_pow2()`

```
template<typename _RandomNumberGenerator>
```

```
int __gnu_parallel::__random_number_pow2 (
 int __logp,
 _RandomNumberGenerator & __rng) [inline]
```

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

|                     |                                                               |
|---------------------|---------------------------------------------------------------|
| <code>__logp</code> | Logarithm (basis 2) of the upper range <code>__bound</code> . |
| <code>__rng</code>  | Random number generator to use.                               |

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

#### `__rd_log2()`

```
template<typename _Size>
_Size __gnu_parallel::__rd_log2 (
 _Size __n) [inline]
```

Calculates the rounded-down logarithm of `__n` for base 2.

#### Parameters

|                        |           |
|------------------------|-----------|
| <code>↔<br/>__n</code> | Argument. |
|------------------------|-----------|

#### Returns

Returns 0 for any argument  $< 1$ .

Referenced by [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>::\\_LoserTreeBase\(\)](#), [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_parallel\\_sort\\_qsb\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [multiseq\\_partition\(\)](#), and [multiseq\\_selection\(\)](#).

#### `__round_up_to_pow2()`

```
template<typename _Tp>
_Tp __gnu_parallel::__round_up_to_pow2 (
 _Tp __x)
```

Round up to the next greater power of 2.

#### Parameters

|                        |                                    |
|------------------------|------------------------------------|
| <code>↔<br/>__x</code> | <code>__Integer</code> to round up |
|------------------------|------------------------------------|

References [\\_\\_rd\\_log2\(\)](#).

Referenced by [\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), and [multiseq\\_selection\(\)](#).

#### `__search_template()`

```
template<typename __RAIter1, typename __RAIter2, typename _Pred>
__RAIter1 __gnu_parallel::__search_template (
 __RAIter1 __begin1,
 __RAIter1 __end1,
 __RAIter2 __begin2,
```

```

 __RAIter2 __end2,
 _Pred __pred)

```

Parallel `std::search`.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__end2</code>   | End iterator of second sequence.   |
| <code>__pred</code>   | Find predicate.                    |

#### Returns

Place of finding in first sequences.

References [\\_\\_calc\\_borders\(\)](#), [\\_\\_equally\\_split\(\)](#), [\\_GLIBCXX\\_CALL](#), [std::max\(\)](#), and [std::min\(\)](#).

#### `__sequential_multiway_merge()`

```

template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↵
_DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::__sequential_multiway_merge (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)

```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

#### Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |
| <code>__sentinel</code>   | The sequences have <code>__a</code> <code>__sentinel</code> element.            |

#### Returns

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_\\_merge\\_advance\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), and [std::min\(\)](#).

Referenced by [multiway\\_merge\(\)](#), [multiway\\_merge\\_sentinels\(\)](#), and [parallel\\_multiway\\_merge\(\)](#).

**\_\_sequential\_random\_shuffle()**

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__sequential_random_shuffle (
 _RAIter __begin,
 _RAIter __end,
 _RandomNumberGenerator & __rng)
```

Sequential cache-efficient random shuffle.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

References [\\_\\_random\\_number\\_pow2\(\)](#), [\\_\\_rd\\_log2\(\)](#), [\\_\\_round\\_up\\_to\\_pow2\(\)](#), [\\_\\_sequential\\_random\\_shuffle\(\)](#), [\\_\\_gnu\\_parallel::Settings::get\(\)](#), [\\_\\_gnu\\_parallel::Settings::L2\\_cache\\_size](#), [std::max\(\)](#), [std::min\(\)](#), [std::partial\\_sum\(\)](#), and [\\_\\_gnu\\_parallel::Settings::TLB\\_size](#).

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

**\_\_shrink()**

```
template<typename _IIter>
void __gnu_parallel::__shrink (
 std::vector<_IIter> & __os_starts,
 size_t & __count_to_two,
 size_t & __range_length)
```

Combines two ranges into one and thus halves the number of ranges.

**Parameters**

|                             |                                          |
|-----------------------------|------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled). |
| <code>__count_to_two</code> | Counts up to 2.                          |
| <code>__range_length</code> | Current length of a chunk.               |

References [std::vector<\\_Tp, \\_Alloc>::size\(\)](#).

Referenced by [\\_\\_shrink\\_and\\_double\(\)](#).

**\_\_shrink\_and\_double()**

```
template<typename _IIter>
void __gnu_parallel::__shrink_and_double (
 std::vector<_IIter> & __os_starts,
 size_t & __count_to_two,
 size_t & __range_length,
 const bool __make_twice)
```

Shrinks and doubles the ranges.

**Parameters**

|                             |                                          |
|-----------------------------|------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled). |
| <code>__count_to_two</code> | Counts up to 2.                          |

|                             |                                                                    |
|-----------------------------|--------------------------------------------------------------------|
| <code>__range_length</code> | Current length of a chunk.                                         |
| <code>__make_twice</code>   | Whether the <code>__os_starts</code> is allowed to be grown or not |

References [\\_\\_shrink\(\)](#), [std::vector<\\_Tp, \\_Alloc>::resize\(\)](#), and [std::vector<\\_Tp, \\_Alloc>::size\(\)](#).  
Referenced by [list\\_partition\(\)](#).

### `__yield()`

```
void __gnu_parallel::__yield () [inline]
```

Yield control to another thread, without waiting for the end of the time slice.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#), and [\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

### `list_partition()`

```
template<typename _IIter, typename _FunctorType>
size_t __gnu_parallel::list_partition (
 const _IIter __begin,
 const _IIter __end,
 _IIter * __starts,
 size_t * __lengths,
 const int __num_parts,
 _FunctorType & __f,
 int __oversampling = 0)
```

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

#### Parameters

|                             |                                                                                                                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>        | Begin iterator of input sequence.                                                                                                                                                                                                                 |
| <code>__end</code>          | End iterator of input sequence.                                                                                                                                                                                                                   |
| <code>__starts</code>       | Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts[__num_parts]</code> contains the end iterator of the sequence.                                                                    |
| <code>__lengths</code>      | Length of the resulting parts.                                                                                                                                                                                                                    |
| <code>__num_parts</code>    | Number of parts to split the sequence into.                                                                                                                                                                                                       |
| <code>__f</code>            | Functor to be applied to each element by traversing <code>__it</code>                                                                                                                                                                             |
| <code>__oversampling</code> | Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num\_parts})$ |

#### Returns

Length of the whole sequence.

References [\\_\\_shrink\\_and\\_double\(\)](#), and [std::vector<\\_Tp, \\_Alloc>::size\(\)](#).

### `max()`

```
template<typename _Tp>
```

```
const _Tp & __gnu_parallel::max (
 const _Tp & __a,
 const _Tp & __b) [inline]
```

Equivalent to `std::max`.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

## min()

```
template<typename _Tp>
const _Tp & __gnu_parallel::min (
 const _Tp & __a,
 const _Tp & __b) [inline]
```

Equivalent to `std::min`.

Referenced by [\\_\\_for\\_each\\_template\\_random\\_access\\_ed\(\)](#), [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\(\)](#), and [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

## multiseq\_partition()

```
template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
void __gnu_parallel::multiseq_partition (
 _RanSeqs __begin_seqs,
 _RanSeqs __end_seqs,
 _RankType __rank,
 _RankIterator __begin_offsets,
 _Compare __comp = std::less< typename std::iterator_traits<typename _RanSeqs>::value_type>::first_type>::value_type>())
```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

### Parameters

|                              |                                                                                                                                                                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code>    | Begin of the sequence of iterator pairs.                                                                                                                                                                                                                     |
| <code>__end_seqs</code>      | End of the sequence of iterator pairs.                                                                                                                                                                                                                       |
| <code>__rank</code>          | The global rank to partition at.                                                                                                                                                                                                                             |
| <code>__begin_offsets</code> | A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> . |
| <code>__comp</code>          | The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .                                                                                                                                                                                        |

References [\\_\\_rd\\_log2\(\)](#), [GLIBCXX\\_CALL](#), [std::distance\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::empty\(\)](#), [std::make\\_pair\(\)](#), [std::max\(\)](#), [std::min\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::pop\(\)](#), [std::priority\\_queue<\\_Tp, \\_Sequence, \\_Compare >::top\(\)](#).

Referenced by [multiway\\_merge\\_exact\\_splitting\(\)](#).

## multiseq\_selection()

```
template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
_Tp __gnu_parallel::multiseq_selection (
 _RanSeqs __begin_seqs,
 _RanSeqs __end_seqs,
 _RankType __rank,
```



```

 __RankType & __offset,
 __Compare __comp = std::less<_Tp>())

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

|                           |                                                                                                                                                            |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code> | Begin of the sequence of iterator pairs.                                                                                                                   |
| <code>__end_seqs</code>   | End of the sequence of iterator pairs.                                                                                                                     |
| <code>__rank</code>       | The global rank to partition at.                                                                                                                           |
| <code>__offset</code>     | The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0. |
| <code>__comp</code>       | The ordering functor, defaults to <code>std::less</code> .                                                                                                 |

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare >::std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare >::top()`.

#### `multiway_merge()`

```

template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
_RAIterOut __gnu_parallel::multiway_merge (
 _RAIterPairIterator __seqs_begin,
 _RAIterPairIterator __seqs_end,
 _RAIterOut __target,
 _DifferenceTp __length,
 _Compare __comp,
 __gnu_parallel::sequential_tag)

```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```

int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 10; ++__j)
 sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;

```

```

for (int __i = 0; __i < 10; ++__i)
{ seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);

```

**See also**

`stable_multiway_merge`

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

**Postcondition**

[\_\_target, return \_\_value) contains merged \_\_elements from the input sequences.

return \_\_value - \_\_target = min(\_\_length, number of elements in all sequences).

**Template Parameters**

|                                  |                                                            |
|----------------------------------|------------------------------------------------------------|
| <code>_RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>_RAIterOut</code>          | iterator over target sequence                              |
| <code>_DifferenceTp</code>       | difference type for the sequence                           |
| <code>_Compare</code>            | strict weak ordering type to compare elements in sequences |

**Parameters**

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | __begin of sequence __sequence                                                  |
| <code>__seqs_end</code>   | __M_end of sequence __sequence                                                  |
| <code>__target</code>     | target sequence to merge to.                                                    |
| <code>__comp</code>       | strict weak ordering to use for element comparison.                             |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

**Returns**

\_\_M\_end iterator of output sequence

References [\\_\\_sequential\\_multiway\\_merge\(\)](#), and [\\_GLIBCXX\\_CALL](#).

**multiway\_merge\_3\_variant()**

```

template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,

```

```

 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)

```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

#### Returns

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

#### `multiway_merge_4_variant()`

```

template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)

```

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

**multiway\_merge\_exact\_splitting()**

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_exact_splitting (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _DifferenceType __length,
 _DifferenceType __total_length,
 _Compare __comp,
 std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

References [\\_\\_equally\\_split\(\)](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), [std::vector< \\_Tp, \\_Alloc >::begin\(\)](#), [std::vector< \\_Tp, \\_Alloc >::end\(\)](#), [multiseq\\_partition\(\)](#), and [std::vector< \\_Tp, \\_Alloc >::resize\(\)](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#).

**multiway\_merge\_loser\_tree()**

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#), and [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#).

**multiway\_merge\_loser\_tree\_sentinel()**

```
template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _↔
DifferenceTp, typename _Compare>
```

```

_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)

```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

### Template Parameters

|                                  |                                                      |
|----------------------------------|------------------------------------------------------|
| <code>_UnguardedLoserTree</code> | Loser Tree variant to use for the unguarded merging. |
|----------------------------------|------------------------------------------------------|

### Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

### Returns

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_GLIBCXX\\_CALL](#), and [multiway\\_merge\\_loser\\_tree\\_unguarded\(\)](#).

### `multiway_merge_loser_tree_unguarded()`

```

template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)

```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

References [\\_GLIBCXX\\_CALL](#).

Referenced by [multiway\\_merge\\_loser\\_tree\\_sentinel\(\)](#).

**multiway\_merge\_sampling\_splitting()**

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_sampling_splitting (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _DifferenceType __length,
 _DifferenceType __total_length,
 _Compare __comp,
 std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.

References [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), and [\\_\\_gnu\\_parallel::\\_Settings::merge\\_oversampling](#).

**multiway\_merge\_sentinels()**

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
 _RAIterPairIterator __seqs_begin,
 _RAIterPairIterator __seqs_end,
 _RAIterOut __target,
 _DifferenceTp __length,
 _Compare __comp,
 __gnu_parallel::sequential_tag)
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable

- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

#### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.

`return __value - __target = min(__length, number of elements in all sequences).`

#### See also

`stable_multiway_merge_sentinels`

#### Template Parameters

|                                  |                                                            |
|----------------------------------|------------------------------------------------------------|
| <code>_RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>_RAIterOut</code>          | iterator over target sequence                              |
| <code>_DifferenceTp</code>       | difference type for the sequence                           |
| <code>_Compare</code>            | strict weak ordering type to compare elements in sequences |

#### Parameters

|                           |                                                          |
|---------------------------|----------------------------------------------------------|
| <code>__seqs_begin</code> | <code>__begin</code> of sequence <code>__sequence</code> |
| <code>__seqs_end</code>   | <code>_M_end</code> of sequence <code>__sequence</code>  |

|                       |                                                                                 |
|-----------------------|---------------------------------------------------------------------------------|
| <code>__target</code> | target sequence to merge to.                                                    |
| <code>__comp</code>   | strict weak ordering to use for element comparison.                             |
| <code>__length</code> | Maximum length to merge, possibly larger than the number of elements available. |

#### Returns

`_M_end` iterator of output sequence

References [\\_\\_sequential\\_multiway\\_merge\(\)](#), and [\\_GLIBCXX\\_CALL](#).

#### **parallel\_multiway\_merge()**

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _↵
_DifferenceTp, typename _Splitter, typename _Compare>
_RAIter3 __gnu_parallel::parallel_multiway_merge (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _Splitter __splitter,
 _DifferenceTp __length,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

#### Template Parameters

|                         |                                                                        |
|-------------------------|------------------------------------------------------------------------|
| <code>_Splitter</code>  | functor to split input (either <code>__exact</code> or sampling based) |
| <code>__stable</code>   | Stable merging incurs a performance penalty.                           |
| <code>__sentinel</code> | Ignored.                                                               |

#### Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

#### Returns

End iterator of output sequence.

References [\\_\\_is\\_sorted\(\)](#), [\\_\\_sequential\\_multiway\\_merge\(\)](#), [\\_GLIBCXX\\_CALL](#), [\\_GLIBCXX\\_PARALLEL\\_LENGTH](#), [\\_\\_gnu\\_parallel::\\_Settings::get\(\)](#), [std::make\\_pair\(\)](#), [\\_\\_gnu\\_parallel::\\_Settings::merge\\_oversampling](#), and [std::min\(\)](#).

Referenced by [\\_\\_parallel\\_merge\\_advance\(\)](#).



**parallel\_sort\_mwms()**

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

PMWMS main call.

**Parameters**

|                            |                             |
|----------------------------|-----------------------------|
| <code>__begin</code>       | Begin iterator of sequence. |
| <code>__end</code>         | End iterator of sequence.   |
| <code>__comp</code>        | Comparator.                 |
| <code>__num_threads</code> | Number of threads to use.   |

References [\\_\\_GLIBCXX\\_CALL](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_pieces](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_samples](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_source](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_starts](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_temporary](#), [\\_\\_gnu\\_parallel::Settings::get\(\)](#), [parallel\\_sort\\_mwms\\_pu\(\)](#), and [\\_\\_gnu\\_parallel::Settings::sort\\_mwms\\_oversampling](#).

Referenced by [\\_\\_parallel\\_sort\(\)](#), [\\_\\_parallel\\_sort\(\)](#), [\\_\\_parallel\\_sort\(\)](#), and [\\_\\_parallel\\_sort\(\)](#).

**parallel\_sort\_mwms\_pu()**

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms_pu (
 _PMWMSortingData<_RAIter> * __sd,
 _Compare & __comp)
```

PMWMS code executed by each thread.

**Parameters**

|                     |                            |
|---------------------|----------------------------|
| <code>__sd</code>   | Pointer to algorithm data. |
| <code>__comp</code> | Comparator.                |

References [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_num\\_threads](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_source](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_starts](#), [\\_\\_gnu\\_parallel::PMWMSortingData<\\_RAIter>::M\\_temporary](#), [\\_\\_gnu\\_parallel::Settings::get\(\)](#), [std::make\\_pair\(\)](#), [\\_\\_gnu\\_parallel::Settings::sort\\_mwms\\_oversampling](#), and [std::uninitialized\\_copy\(\)](#).

Referenced by [parallel\\_sort\\_mwms\(\)](#).

**4.6.5 Variable Documentation****`_CASable_bits`**

```
const int __gnu_parallel::_CASable_bits [static]
Number of bits of _CASable.
```

Referenced by [\\_\\_decode2\(\)](#), and [\\_\\_encode2\(\)](#).

## **`_CASable_mask`**

`const _CASable __gnu_parallel::_CASable_mask [static]`

`_CASable` with the right half of bits set to 1.

Referenced by `__decode2()`.

## **4.7 `__gnu_pbds` Namespace Reference**

### **Classes**

- struct [associative\\_tag](#)
- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)
- struct [binomial\\_heap\\_tag](#)
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)
- struct [container\\_error](#)
- struct [container\\_tag](#)
- struct [container\\_traits](#)
- struct [container\\_traits\\_base](#)
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [container\\_traits\\_base< rb\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [container\\_traits\\_base< thin\\_heap\\_tag >](#)
- class [direct\\_mask\\_range\\_hashing](#)
- class [direct\\_mod\\_range\\_hashing](#)
- class [gp\\_hash\\_table](#)
- struct [gp\\_hash\\_tag](#)
- class [hash\\_exponential\\_size\\_policy](#)
- class [hash\\_load\\_check\\_resize\\_trigger](#)
- class [hash\\_prime\\_size\\_policy](#)
- class [hash\\_standard\\_resize\\_policy](#)
- struct [insert\\_error](#)
- struct [join\\_error](#)
- class [linear\\_probe\\_fn](#)
- class [list\\_update](#)
- struct [list\\_update\\_tag](#)
- class [lu\\_counter\\_policy](#)
- class [lu\\_move\\_to\\_front\\_policy](#)

- struct [null\\_node\\_update](#)
- struct [null\\_type](#)
- struct [ov\\_tree\\_tag](#)
- struct [pairing\\_heap\\_tag](#)
- struct [pat\\_trie\\_tag](#)
- struct [point\\_invalidation\\_guarantee](#)
- class [priority\\_queue](#)
- struct [priority\\_queue\\_tag](#)
- class [quadratic\\_probe\\_fn](#)
- struct [range\\_invalidation\\_guarantee](#)
- struct [rb\\_tree\\_tag](#)
- struct [rc\\_binomial\\_heap\\_tag](#)
- struct [resize\\_error](#)
- class [sample\\_probe\\_fn](#)
- class [sample\\_range\\_hashing](#)
- class [sample\\_ranged\\_hash\\_fn](#)
- class [sample\\_ranged\\_probe\\_fn](#)
- class [sample\\_resize\\_policy](#)
- class [sample\\_resize\\_trigger](#)
- class [sample\\_size\\_policy](#)
- class [sample\\_tree\\_node\\_update](#)
- struct [sample\\_trie\\_access\\_traits](#)
- class [sample\\_trie\\_node\\_update](#)
- struct [sample\\_update\\_policy](#)
- struct [sequence\\_tag](#)
- struct [splay\\_tree\\_tag](#)
- struct [string\\_tag](#)
- struct [thin\\_heap\\_tag](#)
- class [tree](#)
- class [tree\\_order\\_statistics\\_node\\_update](#)
- struct [tree\\_tag](#)
- class [trie](#)
- class [trie\\_order\\_statistics\\_node\\_update](#)
- class [trie\\_prefix\\_search\\_node\\_update](#)
- struct [trie\\_string\\_access\\_traits](#)
- struct [trie\\_tag](#)
- struct [trivial\\_iterator\\_tag](#)

### Typedefs

- typedef void [trivial\\_iterator\\_difference\\_type](#)

### Functions

- void [\\_\\_throw\\_container\\_error](#) ()
- void [\\_\\_throw\\_insert\\_error](#) ()
- void [\\_\\_throw\\_join\\_error](#) ()
- void [\\_\\_throw\\_resize\\_error](#) ()

## Variables

- `template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename ↵_Alloc>  
detail::integral_constant< int, Reverse > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse,  
_Alloc >::s_rev_ind`

### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_sequential` Namespace Reference

### 4.8.1 Detailed Description

GNU sequential classes for public use.

## 4.9 `abi` Namespace Reference

### 4.9.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias 'abi'.

A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.10 `std` Namespace Reference

### Namespaces

- namespace `__debug`
- namespace `__detail`
- namespace `__parallel`
- namespace `chrono`
- namespace `decimal`
- namespace `experimental`
- namespace `filesystem`
- namespace `literals`
- namespace `placeholders`
- namespace `regex_constants`
- namespace `rel_ops`
- namespace `this_thread`
- namespace `tr1`
- namespace `tr2`

## Classes

- class [\\_\\_basic\\_future](#)
- class [\\_\\_codecvt\\_abstract\\_base](#)
- class [\\_\\_ctype\\_abstract\\_base](#)
- struct [\\_\\_is\\_fast\\_hash](#)
- struct [\\_\\_is\\_location\\_invariant](#)
- class [\\_\\_new\\_allocator](#)
- struct [\\_\\_numeric\\_limits\\_base](#)
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- class [\\_Bind](#)
- class [\\_Bind\\_result](#)
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)
- class [\\_Function\\_base](#)
- struct [\\_Fwd\\_list\\_base](#)
- struct [\\_Fwd\\_list\\_const\\_iterator](#)
- struct [\\_Fwd\\_list\\_iterator](#)
- struct [\\_Fwd\\_list\\_node](#)
- struct [\\_Fwd\\_list\\_node\\_base](#)
- class [\\_List\\_base](#)
- struct [\\_List\\_const\\_iterator](#)
- struct [\\_List\\_iterator](#)
- struct [\\_List\\_node](#)
- class [\\_Not\\_fn](#)
- struct [\\_Placeholder](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Tp, false >](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Tp, true >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Vector\\_base](#)
- struct [add\\_const](#)
- struct [add\\_cv](#)
- struct [add\\_lvalue\\_reference](#)
- struct [add\\_pointer](#)
- struct [add\\_rvalue\\_reference](#)
- struct [add\\_volatile](#)
- struct [adopt\\_lock\\_t](#)
- struct [aligned\\_storage](#)
- struct [aligned\\_union](#)
- struct [alignment\\_of](#)
- class [allocator](#)
- class [allocator< void >](#)
- struct [allocator\\_traits](#)
- struct [allocator\\_traits< allocator< \\_Tp > >](#)
- struct [allocator\\_traits< allocator< void > >](#)
- struct [allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >](#)
- struct [array](#)
- struct [atomic](#)
- struct [atomic< \\_Tp \\* >](#)

- struct [atomic< bool >](#)
- struct [atomic< char >](#)
- struct [atomic< char16\\_t >](#)
- struct [atomic< char32\\_t >](#)
- struct [atomic< int >](#)
- struct [atomic< long >](#)
- struct [atomic< long long >](#)
- struct [atomic< short >](#)
- struct [atomic< signed char >](#)
- struct [atomic< unsigned char >](#)
- struct [atomic< unsigned int >](#)
- struct [atomic< unsigned long >](#)
- struct [atomic< unsigned long long >](#)
- struct [atomic< unsigned short >](#)
- struct [atomic< wchar\\_t >](#)
- struct [atomic\\_flag](#)
- class [auto\\_ptr](#)
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)
- class [bad\\_alloc](#)
- class [bad\\_cast](#)
- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)
- class [bad\\_typeid](#)
- class [bad\\_weak\\_ptr](#)
- class [basic\\_filebuf](#)
- class [basic\\_fstream](#)
- class [basic\\_ifstream](#)
- class [basic\\_ios](#)
- class [basic\\_iostream](#)
- class [basic\\_istream](#)
- class [basic\\_istreamstream](#)
- class [basic\\_ofstream](#)
- class [basic\\_ostream](#)
- class [basic\\_ostreamstream](#)
- class [basic\\_regex](#)
- class [basic\\_streambuf](#)
- class [basic\\_string](#)
- class [basic\\_string\\_view](#)
- class [basic\\_stringbuf](#)
- class [basic\\_stringstream](#)
- class [bernoulli\\_distribution](#)
- struct [bidirectional\\_iterator\\_tag](#)
- struct [binary\\_function](#)
- class [binary\\_negate](#)
- class [binder1st](#)
- class [binder2nd](#)
- class [binomial\\_distribution](#)
- class [bitset](#)
- class [cauchy\\_distribution](#)
- struct [char\\_traits](#)

- struct `char_traits< __gnu_cxx::character< _Value, _Int, _St > >`
- struct `char_traits< char >`
- struct `char_traits< wchar_t >`
- class `chi_squared_distribution`
- class `codecvt`
- class `codecvt< _InternT, _ExternT, encoding_state >`
- class `codecvt< char, char, mbstate_t >`
- class `codecvt< char16_t, char, mbstate_t >`
- class `codecvt< char32_t, char, mbstate_t >`
- class `codecvt< wchar_t, char, mbstate_t >`
- class `codecvt_base`
- class `codecvt_byname`
- class `common_iterator`
- struct `common_type`
- struct `common_type< chrono::duration< _Rep, _Period > >`
- struct `common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >`
- struct `common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >`
- struct `common_type< chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >`
- struct `compare_three_way_result`
- class `complex`
- class `complex< double >`
- class `complex< float >`
- class `complex< long double >`
- class `condition_variable`
- class `condition_variable_any`
- struct `conditional`
- class `const_mem_fun1_ref_t`
- class `const_mem_fun1_t`
- class `const_mem_fun_ref_t`
- class `const_mem_fun_t`
- struct `contiguous_iterator_tag`
- class `copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `counted_iterator`
- class `ctype`
- class `ctype< char >`
- class `ctype< wchar_t >`
- struct `ctype_base`
- class `ctype_byname`
- class `ctype_byname< char >`
- struct `default_delete`
- struct `default_delete< _Tp[]>`
- struct `default_sentinel_t`
- struct `defer_lock_t`
- class `deque`
- struct `destroying_delete_t`
- class `discard_block_engine`
- class `discrete_distribution`
- struct `divides`
- struct `divides< void >`

- class `domain_error`
- struct `enable_if`
- class `enable_shared_from_this`
- class `encoding_state`
- struct `equal_to`
- struct `equal_to< void >`
- class `error_category`
- class `error_code`
- class `error_condition`
- class `exception`
- class `exception_ptr`
- class `exponential_distribution`
- struct `extent`
- class `extreme_value_distribution`
- class `fisher_f_distribution`
- struct `forward_iterator_tag`
- class `forward_list`
- class `fpos`
- struct `from_chars_result`
- class `front_insert_iterator`
- class `function< _Res(_ArgTypes...)>`
- class `function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `future`
- class `future< _Res & >`
- class `future< void >`
- class `future_error`
- class `gamma_distribution`
- class `geometric_distribution`
- struct `greater`
- struct `greater< void >`
- struct `greater_equal`
- struct `greater_equal< void >`
- class `gslice`
- class `gslice_array`
- struct `has_virtual_destructor`
- struct `hash`
- struct `hash< __debug::bitset< _Nb > >`
- struct `hash< __debug::vector< bool, _Alloc > >`
- struct `hash< __gnu_cxx::__u16vstring >`
- struct `hash< __gnu_cxx::__u32vstring >`
- struct `hash< __gnu_cxx::__vstring >`
- struct `hash< __gnu_cxx::__wvstring >`
- struct `hash< __gnu_cxx::throw_value_limit >`
- struct `hash< __gnu_cxx::throw_value_random >`
- struct `hash< __gnu_debug::basic_string< _CharT > >`
- struct `hash< __shared_ptr< _Tp, _Lp > >`
- struct `hash< _Tp * >`
- struct `hash< basic_string< char, char_traits< char >, _Alloc > >`
- struct `hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >`
- struct `hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >`
- struct `hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >`



- struct [hash< bool >](#)
- struct [hash< char >](#)
- struct [hash< char16\\_t >](#)
- struct [hash< char32\\_t >](#)
- struct [hash< double >](#)
- struct [hash< error\\_code >](#)
- struct [hash< error\\_condition >](#)
- struct [hash< experimental::optional< \\_Tp > >](#)
- struct [hash< experimental::shared\\_ptr< \\_Tp > >](#)
- struct [hash< float >](#)
- struct [hash< int >](#)
- struct [hash< long >](#)
- struct [hash< long double >](#)
- struct [hash< long long >](#)
- struct [hash< shared\\_ptr< \\_Tp > >](#)
- struct [hash< short >](#)
- struct [hash< signed char >](#)
- struct [hash< thread::id >](#)
- struct [hash< type\\_index >](#)
- struct [hash< unique\\_ptr< \\_Tp, \\_Dp > >](#)
- struct [hash< unsigned char >](#)
- struct [hash< unsigned int >](#)
- struct [hash< unsigned long >](#)
- struct [hash< unsigned long long >](#)
- struct [hash< unsigned short >](#)
- struct [hash< wchar\\_t >](#)
- struct [hash<::bitset< \\_Nb > >](#)
- struct [hash<::vector< bool, \\_Alloc > >](#)
- struct [identity](#)
- class [independent\\_bits\\_engine](#)
- class [indirect\\_array](#)
- class [initializer\\_list](#)
- class [inout\\_ptr\\_t](#)
- struct [input\\_iterator\\_tag](#)
- class [insert\\_iterator](#)
- struct [integer\\_sequence](#)
- struct [integral\\_constant](#)
- class [invalid\\_argument](#)
- class [ios\\_base](#)
- struct [is\\_abstract](#)
- struct [is\\_arithmetic](#)
- struct [is\\_array](#)
- struct [is\\_assignable](#)
- struct [is\\_base\\_of](#)
- struct [is\\_bind\\_expression](#)
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)

- struct [is\\_bind\\_expression](#)< volatile [\\_Bind](#)< [\\_Signature](#) > >
- struct [is\\_bind\\_expression](#)< volatile [\\_Bind\\_result](#)< [\\_Result](#), [\\_Signature](#) > >
- struct [is\\_class](#)
- struct [is\\_compound](#)
- struct [is\\_const](#)
- struct [is\\_constructible](#)
- struct [is\\_copy\\_assignable](#)
- struct [is\\_copy\\_constructible](#)
- struct [is\\_default\\_constructible](#)
- struct [is\\_destructible](#)
- struct [is\\_empty](#)
- struct [is\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)< [future\\_errc](#) >
- struct [is\\_error\\_condition\\_enum](#)
- struct [is\\_floating\\_point](#)
- struct [is\\_function](#)
- struct [is\\_fundamental](#)
- struct [is\\_integral](#)
- struct [is\\_layout\\_compatible](#)
- struct [is\\_literal\\_type](#)
- struct [is\\_lvalue\\_reference](#)
- struct [is\\_member\\_function\\_pointer](#)
- struct [is\\_member\\_object\\_pointer](#)
- struct [is\\_member\\_pointer](#)
- struct [is\\_move\\_assignable](#)
- struct [is\\_move\\_constructible](#)
- struct [is\\_nothrow\\_assignable](#)
- struct [is\\_nothrow\\_constructible](#)
- struct [is\\_nothrow\\_copy\\_assignable](#)
- struct [is\\_nothrow\\_copy\\_constructible](#)
- struct [is\\_nothrow\\_default\\_constructible](#)
- struct [is\\_nothrow\\_destructible](#)
- struct [is\\_nothrow\\_move\\_assignable](#)
- struct [is\\_nothrow\\_move\\_constructible](#)
- struct [is\\_object](#)
- struct [is\\_placeholder](#)
- struct [is\\_placeholder](#)< [\\_Placeholder](#)< [\\_Num](#) > >
- struct [is\\_pod](#)
- struct [is\\_pointer](#)
- struct [is\\_pointer\\_interconvertible\\_base\\_of](#)
- struct [is\\_polymorphic](#)
- struct [is\\_reference](#)
- struct [is\\_rvalue\\_reference](#)
- struct [is\\_same](#)
- struct [is\\_scalar](#)
- struct [is\\_signed](#)
- struct [is\\_standard\\_layout](#)
- struct [is\\_trivial](#)
- struct [is\\_trivially\\_assignable](#)
- struct [is\\_trivially\\_constructible](#)

- struct [is\\_trivially\\_copy\\_assignable](#)
- struct [is\\_trivially\\_copy\\_constructible](#)
- struct [is\\_trivially\\_copyable](#)
- struct [is\\_trivially\\_default\\_constructible](#)
- struct [is\\_trivially\\_destructible](#)
- struct [is\\_trivially\\_move\\_assignable](#)
- struct [is\\_trivially\\_move\\_constructible](#)
- struct [is\\_union](#)
- struct [is\\_unsigned](#)
- struct [is\\_void](#)
- struct [is\\_volatile](#)
- class [istream\\_iterator](#)
- class [istreambuf\\_iterator](#)
- struct [iterator](#)
- struct [iterator\\_traits](#)
- struct [iterator\\_traits< \\_Tp \\* >](#)
- class [length\\_error](#)
- struct [less](#)
- struct [less< void >](#)
- struct [less\\_equal](#)
- struct [less\\_equal< void >](#)
- class [linear\\_congruential\\_engine](#)
- class [list](#)
- class [locale](#)
- class [lock\\_guard](#)
- class [logic\\_error](#)
- struct [logical\\_and](#)
- struct [logical\\_and< void >](#)
- struct [logical\\_not](#)
- struct [logical\\_not< void >](#)
- struct [logical\\_or](#)
- struct [logical\\_or< void >](#)
- class [lognormal\\_distribution](#)
- struct [make\\_signed](#)
- struct [make\\_unsigned](#)
- class [map](#)
- class [mask\\_array](#)
- class [match\\_results](#)
- class [mem\\_fun1\\_ref\\_t](#)
- class [mem\\_fun1\\_t](#)
- class [mem\\_fun\\_ref\\_t](#)
- class [mem\\_fun\\_t](#)
- class [mersenne\\_twister\\_engine](#)
- class [messages](#)
- struct [messages\\_base](#)
- class [messages\\_byname](#)
- struct [minus](#)
- struct [minus< void >](#)
- struct [modulus](#)
- struct [modulus< void >](#)
- class [money\\_base](#)

- class `money_get`
- class `money_put`
- class `money_punct`
- class `money_punct_byname`
- class `move_iterator`
- class `move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`
- class `move_sentinel`
- class `multimap`
- struct `multiplies`
- struct `multiplies< void >`
- class `multiset`
- class `mutex`
- struct `negate`
- struct `negate< void >`
- class `negative_binomial_distribution`
- class `nested_exception`
- class `normal_distribution`
- struct `not_equal_to`
- struct `not_equal_to< void >`
- class `num_get`
- class `num_put`
- struct `numeric_limits`
- struct `numeric_limits< bool >`
- struct `numeric_limits< char >`
- struct `numeric_limits< char16_t >`
- struct `numeric_limits< char32_t >`
- struct `numeric_limits< double >`
- struct `numeric_limits< float >`
- struct `numeric_limits< int >`
- struct `numeric_limits< long >`
- struct `numeric_limits< long double >`
- struct `numeric_limits< long long >`
- struct `numeric_limits< short >`
- struct `numeric_limits< signed char >`
- struct `numeric_limits< unsigned char >`
- struct `numeric_limits< unsigned int >`
- struct `numeric_limits< unsigned long >`
- struct `numeric_limits< unsigned long long >`
- struct `numeric_limits< unsigned short >`
- struct `numeric_limits< wchar_t >`
- class `num_punct`
- class `num_punct_byname`
- struct `once_flag`
- class `ostream_iterator`
- class `ostreambuf_iterator`
- class `out_of_range`
- class `out_ptr_t`
- struct `output_iterator_tag`
- class `overflow_error`
- struct `owner_less`
- struct `owner_less< shared_ptr< _Tp > >`

- struct [owner\\_less< void >](#)
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)
- class [packaged\\_task< \\_Res\(\\_ArgTypes...\)>](#)
- struct [pair](#)
- class [philox\\_engine](#)
- class [piecewise\\_constant\\_distribution](#)
- struct [piecewise\\_construct\\_t](#)
- class [piecewise\\_linear\\_distribution](#)
- struct [plus](#)
- class [pointer\\_to\\_binary\\_function](#)
- class [pointer\\_to\\_unary\\_function](#)
- struct [pointer\\_traits](#)
- struct [pointer\\_traits< \\_Tp \\* >](#)
- class [poisson\\_distribution](#)
- class [priority\\_queue](#)
- class [promise](#)
- class [promise< \\_Res & >](#)
- class [promise< void >](#)
- class [queue](#)
- struct [random\\_access\\_iterator\\_tag](#)
- class [random\\_device](#)
- class [range\\_error](#)
- struct [rank](#)
- struct [ratio](#)
- struct [ratio\\_equal](#)
- struct [ratio\\_greater](#)
- struct [ratio\\_greater\\_equal](#)
- struct [ratio\\_less](#)
- struct [ratio\\_less\\_equal](#)
- struct [ratio\\_not\\_equal](#)
- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)
- class [recursive\\_timed\\_mutex](#)
- class [reference\\_wrapper](#)
- class [regex\\_error](#)
- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- class [regex\\_traits](#)
- struct [remove\\_all\\_extents](#)
- struct [remove\\_const](#)
- struct [remove\\_cv](#)
- struct [remove\\_extent](#)
- struct [remove\\_pointer](#)
- struct [remove\\_reference](#)
- struct [remove\\_volatile](#)
- struct [result\\_of](#)
- class [reverse\\_iterator](#)
- class [runtime\\_error](#)
- class [scoped\\_allocator\\_adaptor](#)
- class [seed\\_seq](#)
- class [set](#)

- class `shared_future`
- class `shared_future< _Res & >`
- class `shared_future< void >`
- class `shared_lock`
- class `shared_ptr`
- class `shared_timed_mutex`
- class `shuffle_order_engine`
- class `slice`
- class `slice_array`
- class `stack`
- class `student_t_distribution`
- class `sub_match`
- class `subtract_with_carry_engine`
- class `system_error`
- class `thread`
- class `time_base`
- class `time_get`
- class `time_get_byname`
- class `time_put`
- class `time_put_byname`
- class `timed_mutex`
- struct `to_chars_result`
- struct `try_to_lock_t`
- class `tuple`
- class `tuple< _T1, _T2 >`
- struct `tuple_element`
- struct `tuple_element< 0, pair< _Tp1, _Tp2 > >`
- struct `tuple_element< 1, pair< _Tp1, _Tp2 > >`
- struct `tuple_element< __i, tuple< _Types... > >`
- struct `tuple_element< _Ind, array< _Tp, _Nm > >`
- struct `tuple_size`
- struct `tuple_size< array< _Tp, _Nm > >`
- struct `tuple_size< pair< _Tp1, _Tp2 > >`
- struct `tuple_size< tuple< _Elements... > >`
- struct `type_index`
- class `type_info`
- struct `type_order`
- struct `unary_function`
- class `unary_negate`
- class `underflow_error`
- struct `underlying_type`
- class `uniform_int_distribution`
- class `uniform_real_distribution`
- class `unique_lock`
- class `unique_ptr`
- class `unique_ptr< _Tp[], _Dp >`
- class `unordered_map`
- class `unordered_multimap`
- class `unordered_multiset`
- class `unordered_set`
- struct `uses_allocator`

- struct `uses_allocator< tuple< _Types... >, _Alloc >`
- class `valarray`
- class `vector`
- class `vector< bool, _Alloc >`
- class `wbuffer_convert`
- class `weak_ptr`
- class `weibull_distribution`
- class `wstring_convert`

## Concepts

- concept `assignable_from`
- concept `common_reference_with`
- concept `common_with`
- concept `constructible_from`
- concept `convertible_to`
- concept `copy_constructible`
- concept `default_initializable`
- concept `derived_from`
- concept `destructible`
- concept `equivalence_relation`
- concept `indirectly_comparable`
- concept `indirectly_copyable`
- concept `indirectly_movable`
- concept `indirectly_readable`
- concept `indirectly_swappable`
- concept `indirectly_writable`
- concept `invocable`
- concept `mergeable`
- concept `move_constructible`
- concept `permutable`
- concept `predicate`
- concept `regular_invocable`
- concept `relation`
- concept `same_as`
- concept `sortable`
- concept `strict_weak_order`
- concept `uniform_random_bit_generator`
- concept `weakly_incrementable`

## Typedefs

- `template<typename _Tp>`  
  using `__allocator_base`
- `typedef unsigned char __atomic_flag_data_type`
- `typedef FILE __c_file`
- `typedef __locale_t __c_locale`
- `typedef __gthread_mutex_t __c_lock`
- `template<bool _Cond, typename _If, typename _Else>`  
  using `__conditional_t`

- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::value>::type, size_t = tuple_size<_Tp>::value>`  
`using __enable_if_has_tuple_size`
- `template<bool _Cond, typename _Tp = void>`  
`using __enable_if_t`
- `template<typename _Fn, typename _Op>`  
`using __function_guide_t`
- `template<typename _Func, typename _SfinaeType>`  
`using __has_is_transparent_t`
- `template<typename _Tp>`  
`using __indirect_value_t`
- `template<typename _ToElementType, typename _FromElementType>`  
`using __is_array_convertible`
- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
`using __is_socketlike`
- `template<typename _InputIterator>`  
`using __iter_key_t`
- `template<typename _InputIterator>`  
`using __iter_to_alloc_t`
- `template<typename _InputIterator>`  
`using __iter_val_t`
- `template<typename _Tp, typename _Up>`  
`using __like_t`
- `template<typename _Ptr, typename _Tp>`  
`using __ptr_rebind`
- `template<typename _Is, typename _Tp>`  
`using __rvalue_stream_extraction_t`
- `template<typename _Os, typename _Tp>`  
`using __rvalue_stream_insertion_t`
- `typedef basic_string< char > __sso_string`
- `template<size_t __i, typename _Tp>`  
`using __tuple_element_t`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc =`  
`std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __umap_hashtable`
- `template<bool _Cache>`  
`using __umap_traits`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc =`  
`std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __ummap_hashtable`
- `template<bool _Cache>`  
`using __ummap_traits`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::`  
`allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __umset_hashtable`
- `template<bool _Cache>`  
`using __umset_traits`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::`  
`allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __uset_hashtable`
- `template<bool _Cache>`  
`using __uset_traits`



- `template<typename _Fn, typename... _Args>`  
  `using \_Bind\_back\_t`
- `template<typename _Fn, typename... _Args>`  
  `using \_Bind\_front\_t`
- `typedef unsigned long \_Bit\_type`
- `template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>`  
  `using \_If\_fs\_path`
- `template<typename _Tp>`  
  `using add\_lvalue\_reference\_t`
- `template<typename _Tp>`  
  `using add\_pointer\_t`
- `template<typename _Tp>`  
  `using add\_rvalue\_reference\_t`
- `template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>`  
  `using aligned\_storage\_t`
- `template<size_t _Len, typename... _Types>`  
  `using aligned\_union\_t`
- `typedef atomic< bool > atomic\_bool`
- `typedef atomic< char > atomic\_char`
- `typedef atomic< char16_t > atomic\_char16\_t`
- `typedef atomic< char32_t > atomic\_char32\_t`
- `typedef atomic< int > atomic\_int`
- `typedef atomic< int16_t > atomic\_int16\_t`
- `typedef atomic< int32_t > atomic\_int32\_t`
- `typedef atomic< int64_t > atomic\_int64\_t`
- `typedef atomic< int8_t > atomic\_int8\_t`
- `typedef atomic< int_fast16_t > atomic\_int\_fast16\_t`
- `typedef atomic< int_fast32_t > atomic\_int\_fast32\_t`
- `typedef atomic< int_fast64_t > atomic\_int\_fast64\_t`
- `typedef atomic< int_fast8_t > atomic\_int\_fast8\_t`
- `typedef atomic< int_least16_t > atomic\_int\_least16\_t`
- `typedef atomic< int_least32_t > atomic\_int\_least32\_t`
- `typedef atomic< int_least64_t > atomic\_int\_least64\_t`
- `typedef atomic< int_least8_t > atomic\_int\_least8\_t`
- `typedef atomic< intmax_t > atomic\_intmax\_t`
- `typedef atomic< intptr_t > atomic\_intptr\_t`
- `typedef atomic< long long > atomic\_llong`
- `typedef atomic< long > atomic\_long`
- `typedef atomic< ptrdiff_t > atomic\_ptrdiff\_t`
- `typedef atomic< signed char > atomic\_schar`
- `typedef atomic< short > atomic\_short`
- `typedef atomic< size_t > atomic\_size\_t`
- `typedef atomic< unsigned char > atomic\_uchar`
- `typedef atomic< unsigned int > atomic\_uint`
- `typedef atomic< uint16_t > atomic\_uint16\_t`
- `typedef atomic< uint32_t > atomic\_uint32\_t`
- `typedef atomic< uint64_t > atomic\_uint64\_t`
- `typedef atomic< uint8_t > atomic\_uint8\_t`
- `typedef atomic< uint_fast16_t > atomic\_uint\_fast16\_t`
- `typedef atomic< uint_fast32_t > atomic\_uint\_fast32\_t`
- `typedef atomic< uint_fast64_t > atomic\_uint\_fast64\_t`

- typedef [atomic](#)< uint\_fast8\_t > [atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [atomic\\_wchar\\_t](#)
- using **atto**
- using **centi**
- typedef [match\\_results](#)< const char \* > **cmatch**
- template<typename... \_Ts>  
using **common\_comparison\_category\_t**
- template<typename... \_Tp>  
using [common\\_type\\_t](#)
- template<typename \_Tp, typename \_Up = \_Tp>  
using [compare\\_three\\_way\\_result\\_t](#)
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse>  
using [conditional\\_t](#)
- template<input\_iterator \_It>  
using **const\_iterator**
- template<semiregular \_Sent>  
using **const\_sentinel**
- typedef [regex\\_iterator](#)< const char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- using **deca**
- template<typename \_Tp>  
using [decay\\_t](#)
- using **deci**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- template<typename \_IndexType, size\_t \_Rank>  
using **dextents**
- template<size\_t \_Rank, typename \_IndexType = size\_t>  
using **dims**
- template<bool \_Cond, typename \_Tp = void>  
using [enable\\_if\\_t](#)
- using **exa**
- using [false\\_type](#)
- using **femto**
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)
- using **giga**
- using **hecto**
- typedef [basic\\_ifstream](#)< char > [ifstream](#)
- template<size\_t... \_Idx>  
using [index\\_sequence](#)
- template<typename... \_Types>  
using [index\\_sequence\\_for](#)

- `template<typename _Fn, typename... _Is>`  
  using **indirect\_result\_t**
- `typedef basic_ios< char > ios`
- `typedef basic_iostream< char > iostream`
- `typedef basic_istream< char > istream`
- `typedef basic_istreamstream< char > istringstream`
- `template<indirectly_readable _Tp>`  
  using **iter\_common\_reference\_t**
- `template<indirectly_readable _It>`  
  using **iter\_const\_reference\_t**
- `template<typename _Tp>`  
  using **iter\_difference\_t**
- `template<__detail::__dereferenceable _Tp>`  
  using **iter\_reference\_t**
- `template<__detail::__dereferenceable _Tp>`  
  using **iter\_rvalue\_reference\_t**
- `template<typename _Tp>`  
  using **iter\_value\_t**
- using **kilo**
- `typedef shuffle_order_engine< minstd_rand0, 256 > knuth_b`
- `template<size_t _Num>`  
  using **make\_index\_sequence**
- `template<typename _Tp, _Tp _Num>`  
  using **make\_integer\_sequence**
- `template<typename _Tp>`  
  using **make\_signed\_t**
- `template<typename _Tp>`  
  using **make\_unsigned\_t**
- using **mega**
- using **micro**
- using **milli**
- `typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand`
- `typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0`
- `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937`
- `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffffeee000000000ULL, 43, 6364136223846793005ULL > mt19937_64`
- using **nano**
- `typedef void(* new_handler) ()`
- `typedef decltype(nullptr) nullptr_t`
- `typedef basic_ofstream< char > ofstream`
- `typedef basic_ostream< char > ostream`
- `typedef basic_ostringstream< char > ostringstream`
- using **peta**
- `typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > philox4x32`
- `typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470EE14C6C93, 0xBB67AE8584CAA73B > philox4x64`
- using **pico**
- `template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>`  
  using **projected**

---

- `template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>`  
`using projected_value_t`
- `typedef __PTRDIFF_TYPE__ ptrdiff_t`
- `typedef discard\_block\_engine< ranlux24_base, 223, 23 > ranlux24`
- `typedef subtract\_with\_carry\_engine< uint_fast32_t, 24, 10, 24 > ranlux24_base`
- `typedef discard\_block\_engine< ranlux48_base, 389, 11 > ranlux48`
- `typedef subtract\_with\_carry\_engine< uint_fast64_t, 48, 5, 12 > ranlux48_base`
- `template<typename _R1, typename _R2>`  
`using ratio\_add`
- `template<typename _R1, typename _R2>`  
`using ratio\_divide`
- `template<typename _R1, typename _R2>`  
`using ratio\_multiply`
- `template<typename _R1, typename _R2>`  
`using ratio\_subtract`
- `typedef basic\_regex< char > regex`
- `template<typename _Tp>`  
`using remove\_all\_extents\_t`
- `template<typename _Tp>`  
`using remove\_extent\_t`
- `template<typename _Tp>`  
`using remove\_pointer\_t`
- `template<typename _Tp>`  
`using remove\_reference\_t`
- `template<typename _Tp>`  
`using result\_of\_t`
- `typedef __SIZE_TYPE__ size_t`
- `typedef match\_results< string::const_iterator > smatch`
- `typedef regex\_iterator< string::const_iterator > sregex_iterator`
- `typedef regex\_token\_iterator< string::const_iterator > sregex_token_iterator`
- `typedef sub\_match< string::const_iterator > ssub_match`
- `typedef basic\_streambuf< char > streambuf`
- `typedef long long streamoff`
- `typedef fpos< mbstate_t > streampos`
- `typedef ptrdiff_t streamsize`
- `typedef basic\_string< char > string`
- `using string_view`
- `typedef basic\_stringbuf< char > stringbuf`
- `typedef basic\_stringstream< char > stringstream`
- `using tera`
- `typedef void(* terminate\_handler) ()`
- `using true\_type`
- `template<size_t __i, typename _Tp>`  
`using tuple_element_t`
- `typedef fpos< mbstate_t > u16streampos`
- `typedef basic\_string< char16_t > u16string`
- `using u16string_view`
- `typedef fpos< mbstate_t > u32streampos`
- `typedef basic\_string< char32_t > u32string`
- `using u32string_view`
- `template<typename _Tp>`  
`using underlying\_type\_t`

---

- typedef [match\\_results](#)< const wchar\_t \* > **wcmatch**
- typedef [regex\\_iterator](#)< const wchar\_t \* > **wcregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const wchar\_t \* > **wcregex\_token\_iterator**
- typedef [sub\\_match](#)< const wchar\_t \* > **wcsub\_match**
- typedef [basic\\_filebuf](#)< wchar\_t > **wfilebuf**
- typedef [basic\\_fstream](#)< wchar\_t > **wfstream**
- typedef [basic\\_ifstream](#)< wchar\_t > **wifstream**
- typedef [basic\\_ios](#)< wchar\_t > **wios**
- typedef [basic\\_iostream](#)< wchar\_t > **wiostream**
- typedef [basic\\_istream](#)< wchar\_t > **wistream**
- typedef [basic\\_istreamstream](#)< wchar\_t > **wistreamstream**
- typedef [basic\\_ofstream](#)< wchar\_t > **wofstream**
- typedef [basic\\_ostream](#)< wchar\_t > **wostream**
- typedef [basic\\_ostreamstream](#)< wchar\_t > **wostringstream**
- typedef [basic\\_regex](#)< wchar\_t > **wregex**
- typedef [match\\_results](#)< wstring::const\_iterator > **wsmatch**
- typedef [regex\\_iterator](#)< wstring::const\_iterator > **wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > **wsregex\_token\_iterator**
- typedef [sub\\_match](#)< wstring::const\_iterator > **wssub\_match**
- typedef [basic\\_streambuf](#)< wchar\_t > **wstreambuf**
- typedef [fpos](#)< mbstate\_t > **wstreampos**
- typedef [basic\\_string](#)< wchar\_t > **wstring**
- using **wstring\_view**
- typedef [basic\\_stringbuf](#)< wchar\_t > **wstringbuf**
- typedef [basic\\_stringstream](#)< wchar\_t > **wstringstream**

## Enumerations

- enum { **\_S\_chunk\_size** }
- enum { **\_S\_word\_bit** }
- enum **\_ios\_Fmtflags** {  
**\_S\_boolalpha** , **\_S\_dec** , **\_S\_fixed** , **\_S\_hex** ,  
**\_S\_internal** , **\_S\_left** , **\_S\_oct** , **\_S\_right** ,  
**\_S\_scientific** , **\_S\_showbase** , **\_S\_showpoint** , **\_S\_showpos** ,  
**\_S\_skipws** , **\_S\_unitbuf** , **\_S\_uppercase** , **\_S\_adjustfield** ,  
**\_S\_basefield** , **\_S\_floatfield** , **\_S\_ios\_fmtflags\_end** , **\_S\_ios\_fmtflags\_max** ,  
**\_S\_ios\_fmtflags\_min** }
- enum **\_ios\_iostate** {  
**\_S\_goodbit** , **\_S\_badbit** , **\_S\_eofbit** , **\_S\_failbit** ,  
**\_S\_ios\_iostate\_end** , **\_S\_ios\_iostate\_max** , **\_S\_ios\_iostate\_min** }
- enum **\_ios\_Openmode** {  
**\_S\_app** , **\_S\_ate** , **\_S\_bin** , **\_S\_in** ,  
**\_S\_out** , **\_S\_trunc** , **\_S\_noreplace** , **\_S\_ios\_openmode\_end** ,  
**\_S\_ios\_openmode\_max** , **\_S\_ios\_openmode\_min** }
- enum **\_ios\_Seekdir** { **\_S\_beg** , **\_S\_cur** , **\_S\_end** , **\_S\_ios\_seekdir\_end** }
- enum **\_Lock\_policy**
- enum **\_Manager\_operation** { **\_\_get\_type\_info** , **\_\_get\_func\_ptr** , **\_\_clone\_func** , **\_\_destroy\_func** }
- enum **\_Rb\_tree\_color** { **\_S\_red** , **\_S\_black** }
- enum class **align\_val\_t** : size\_t
- enum class **chars\_format** { **scientific** , **fixed** , **hex** , **general** }
- enum **codecvt\_mode** { **consume\_header** , **generate\_header** , **little\_endian** }
- enum class **cv\_status** { **no\_timeout** , **timeout** }

- enum class **errc** {  
**address\_family\_not\_supported** , **address\_in\_use** , **address\_not\_available** , **already\_connected** ,  
**argument\_list\_too\_long** , **argument\_out\_of\_domain** , **bad\_address** , **bad\_file\_descriptor** ,  
**broken\_pipe** , **connection\_aborted** , **connection\_already\_in\_progress** , **connection\_refused** ,  
**connection\_reset** , **cross\_device\_link** , **destination\_address\_required** , **device\_or\_resource\_busy** ,  
**directory\_not\_empty** , **executable\_format\_error** , **file\_exists** , **file\_too\_large** ,  
**filename\_too\_long** , **function\_not\_supported** , **host\_unreachable** , **illegal\_byte\_sequence** ,  
**inappropriate\_io\_control\_operation** , **interrupted** , **invalid\_argument** , **invalid\_seek** ,  
**io\_error** , **is\_a\_directory** , **message\_size** , **network\_down** ,  
**network\_reset** , **network\_unreachable** , **no\_buffer\_space** , **no\_child\_process** ,  
**no\_lock\_available** , **no\_message** , **no\_protocol\_option** , **no\_space\_on\_device** ,  
**no\_such\_device\_or\_address** , **no\_such\_device** , **no\_such\_file\_or\_directory** , **no\_such\_process** ,  
**not\_a\_directory** , **not\_a\_socket** , **not\_connected** , **not\_enough\_memory** ,  
**operation\_in\_progress** , **operation\_not\_permitted** , **operation\_not\_supported** , **operation\_would\_block** ,  
**permission\_denied** , **protocol\_not\_supported** , **read\_only\_file\_system** , **resource\_deadlock\_would\_occur** ,  
**resource\_unavailable\_try\_again** , **result\_out\_of\_range** , **timed\_out** , **too\_many\_files\_open\_in\_system** ,  
**too\_many\_files\_open** , **too\_many\_links** , **too\_many\_symbolic\_link\_levels** , **wrong\_protocol\_type** }
- enum **float\_denorm\_style** { **denorm\_indeterminate** , **denorm\_absent** , **denorm\_present** }
- enum **float\_round\_style** {  
**round\_indeterminate** , **round\_toward\_zero** , **round\_to\_nearest** , **round\_toward\_infinity** ,  
**round\_toward\_neg\_infinity** }
- enum class **future\_errc** { **future\_already\_retrieved** , **promise\_already\_satisfied** , **no\_state** , **broken\_promise** }
- enum class **future\_status** { **ready** , **timeout** , **deferred** }
- enum class **io\_errc** { **stream** }
- enum class **launch** { **async** , **deferred** }
- enum class **memory\_order** : int {  
**relaxed** , **consume** , **acquire** , **release** ,  
**acq\_rel** , **seq\_cst** }

## Functions

- template<typename \_CharT>  
**\_CharT \* \_\_add\_gathering** (\_CharT \*\_\_s, \_CharT \_\_sep, const char \*\_\_gbeg, size\_t \_\_gsize, const \_CharT \*\_\_first, const \_CharT \*\_\_last)
- template<typename \_Tp>  
**constexpr \_Tp \* \_\_addressof** (\_Tp &\_\_r) noexcept
- template<typename \_ForwardIterator, typename \_BinaryPredicate>  
**constexpr \_ForwardIterator \_\_adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Tp, typename \_Compare>  
**constexpr void \_\_adjust\_heap** (\_RandomAccessIterator \_\_first, \_Distance \_\_holeIndex, \_Distance \_\_len, \_Tp \_\_value, \_Compare \_\_comp)
- template<typename \_Tp, \_Lock\_policy \_Lp = \_\_default\_lock\_policy, typename \_Alloc, typename... \_Args>  
**\_\_shared\_ptr< \_Tp, \_Lp > \_\_allocate\_shared** (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- template<bool \_IsMove, typename \_OutIter, typename \_InIter>  
**constexpr void \_\_assign\_one** (\_OutIter &\_\_out, \_InIter &\_\_in)
- template<typename \_Facet>  
**const \_Facet & \_\_check\_facet** (const \_Facet \*\_\_f)
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Compare>  
**constexpr void \_\_chunk\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Distance \_\_chunk\_size, \_Compare \_\_comp)

- `template<typename _Tp>`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `__complex__ double __complex_acos (__complex__ double __z)`
- `__complex__ float __complex_acos (__complex__ float __z)`
- `__complex__ long double __complex_acos (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_acosh (__complex__ double __z)`
- `__complex__ float __complex_acosh (__complex__ float __z)`
- `__complex__ long double __complex_acosh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `__complex__ double __complex_asin (__complex__ double __z)`
- `__complex__ float __complex_asin (__complex__ float __z)`
- `__complex__ long double __complex_asin (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_asinh (__complex__ double __z)`
- `__complex__ float __complex_asinh (__complex__ float __z)`
- `__complex__ long double __complex_asinh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_atan (__complex__ double __z)`
- `__complex__ float __complex_atan (__complex__ float __z)`
- `__complex__ long double __complex_atan (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `__complex__ double __complex_atanh (__complex__ double __z)`
- `__complex__ float __complex_atanh (__complex__ float __z)`
- `__complex__ long double __complex_atanh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `complex< double > __complex_proj (const complex< double > &__z)`
- `complex< float > __complex_proj (const complex< float > &__z)`
- `complex< long double > __complex_proj (const complex< long double > &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`

- `template<typename _Tp>`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)`
- `template<typename _Tp>`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<> void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > __copy_move_a (_II, _II, const __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr __gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > __copy_move_a (const __gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const __gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const __gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI __copy_move_a (const __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`__gnu_debug::Safe_iterator< _OTp, _OTp &, _OTp * > __copy_move_a1 (__gnu_debug::Safe_iterator< _ITp, _IRef, _IPtr > __first, __gnu_debug::Safe_iterator< _ITp, _IRef, _IPtr > __last, __gnu_debug::Safe_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI __copy_move_a1 (__gnu_debug::Safe_iterator< _Tp, _Ref, _Ptr > __first, __gnu_debug::Safe_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`__gnu_cxx::enable_if< __is_any_random_access_iter< _II >::__value, __gnu_debug::Safe_iterator< _Tp, _Tp &, _Tp * > >::__type __copy_move_a1 (_II __first, _II __last, __gnu_debug::Safe_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI __copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>`  
`constexpr _OutIter __copy_move_a2 (_InIter __first, _Sent __last, _OutIter __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`



- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT> >::__type __copy_↵`  
`__move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator<`  
`_CharT> __first, istreambuf_iterator<_CharT> __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ::Deque_iterator<_CharT, _CharT &, _↵`  
`CharT * > >::__type __copy_move_a2 (istreambuf_iterator<_CharT, char_traits<_CharT> > __first,`  
`istreambuf_iterator<_CharT, char_traits<_CharT> > __last, ::Deque_iterator<_CharT, _CharT &, _CharT *`  
`> __result)`
- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator<`  
`_CharT, char_traits<_CharT> >, istreambuf_iterator<_CharT, char_traits<_CharT> >, _CharT *)`
- `template<bool _IsMove, typename _Il, typename _OI>`  
`constexpr _OI __copy_move_backward_a (_Il __first, _Il __last, _OI __result)`
- `template<bool _IsMove, typename _Il, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr __gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> __copy_move_backward_a (_Il, _Il, const`  
`::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &)`
- `template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr ::__gnu_debug::Safe_iterator<_OIte, _OSeq, _OCat> __copy_move_backward_a (const`  
`::__gnu_debug::Safe_iterator<_Ite, _ISeq, _ICat> &, const ::__gnu_debug::Safe_iterator<_OIte, _ISeq,`  
`_ICat> &, const ::__gnu_debug::Safe_iterator<_OIte, _OSeq, _OCat> &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI __copy_move_backward_a (const ::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &, const`  
`::__gnu_debug::Safe_iterator<_Ite, _Seq, _Cat> &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`::Deque_iterator<_OTp, _OTp &, _OTp * > __copy_move_backward_a1 (::Deque_iterator<_ITp, _IRef,`  
`_IPtr> __first, ::Deque_iterator<_ITp, _IRef, _IPtr> __last, ::Deque_iterator<_OTp, _OTp &, _OTp * >`  
`__result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI __copy_move_backward_a1 (::Deque_iterator<_Tp, _Ref, _Ptr> __first, ::Deque_iterator<_Tp, _Ref,`  
`_Ptr> __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 __copy_move_backward_a1 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _Il, typename _Tp>`  
`__gnu_cxx::__enable_if<__is_any_random_access_iter<_Il>::__value, ::Deque_iterator<_Tp, _Tp &, _Tp`  
`* > >::__type __copy_move_backward_a1 (_Il __first, _Il __last, ::Deque_iterator<_Tp, _Tp &, _Tp * >`  
`__result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI __copy_move_backward_dit (::Deque_iterator<_Tp, _Ref, _Ptr> __first, ::Deque_iterator<_Tp, _Ref,`  
`_Ptr> __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI __copy_move_dit (::Deque_iterator<_Tp, _Ref, _Ptr> __first, ::Deque_iterator<_Tp, _Ref, _Ptr >`  
`__last, _OI __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator __copy_n_a (_InputIterator __first, _Size __n, _OutputIterator __result, bool)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type __copy_n_a (istreambuf_iterator<`  
`_CharT> __it, _Size __n, _CharT * __result, bool __strict)`

- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, __Deque_iterator< _CharT, _CharT &, _CharT`  
`* > >::__type __copy_n_a (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size,`  
`::__Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_n_a (istreambuf_iterator<`  
`_CharT, char_traits< _CharT > >, _Size, _CharT *, bool)`
- `template<typename _CharT, typename _Traits>`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _`  
`_Traits > *__sout)`
- `template<typename _CharT, typename _Traits>`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT,`  
`_Traits > *__sout, bool &__ineof)`
- `template<> streamsize __copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_streambuf< char`  
`> *__sout, bool &__ineof)`
- `template<> streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf<`  
`wchar_t > *__sout, bool &__ineof)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type __count_if (_InputIterator __first, _InputIterator __`  
`last, _Predicate __pred)`
- `constexpr size_t __deque_buf_size (size_t __size)`
- `template<typename _Tp>`  
`ptrdiff_t __distance (::__List_const_iterator< _Tp >, __List_const_iterator< _Tp >, input_iterator_tag)`
- `template<typename _Tp>`  
`ptrdiff_t __distance (::__List_iterator< _Tp >, __List_iterator< _Tp >, input_iterator_tag)`
- `template<bool _Const, typename _Ptr>`  
`ptrdiff_t __distance (__list::__Iterator< _Const, _Ptr > __first, __list::__Iterator< _Const, _Ptr > __last,`  
`input_iterator_tag __tag)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::difference_type __distance (_InputIterator __first, _InputIterator __`  
`last, input_iterator_tag)`
- `template<typename _OutputIterator>`  
`void __distance (_OutputIterator, _OutputIterator, output_iterator_tag)=delete`
- `template<typename _RandomAccessIterator>`  
`constexpr iterator_traits< _RandomAccessIterator >::difference_type __distance (_RandomAccessIterator __`  
`first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _I1, typename _I2>`  
`constexpr bool __equal4 (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _BinaryPredicate>`  
`constexpr bool __equal4 (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2>`  
`constexpr bool __equal_aux (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool __equal_aux (_I1, _I1, const ::__gnu_debug::__Safe_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool __equal_aux (const ::__gnu_debug::__Safe_iterator< _I1, _Seq1, _Cat1 > &, const`  
`::__gnu_debug::__Safe_iterator< _I1, _Seq1, _Cat1 > &, _I2)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool __equal_aux (const ::__gnu_debug::__Safe_iterator< _I1, _Seq1, _Cat1 > &, const`  
`::__gnu_debug::__Safe_iterator< _I1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _I2, _Seq2,`  
`_Cat2 > &)`

- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>  
__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type __equal_aux1  
(::Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>  
bool __equal_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1  
> __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr>  
__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type __equal_aux1 (_II <-  
__first1, _II __last1, ::Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _II1, typename _II2>  
constexpr bool __equal_aux1 (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>  
bool __equal_dit (const ::Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::Deque_iterator< _Tp, _Ref,  
_Ptr > &__last1, _II __first2)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>  
constexpr pair< _ForwardIterator, _ForwardIterator > __equal_range (_ForwardIterator __first, _ForwardIterator  
__last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Up = _Tp>  
constexpr _Tp __exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _FIter, typename _Tp>  
constexpr void __fill_a (_FIter __first, _FIter __last, const _Tp &__value)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>  
constexpr void __fill_a (const ::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const ::gnu_debug::Safe_iterator<  
_Ite, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Ite, typename _Cont, typename _Tp>  
constexpr void __fill_a1 (::gnu_cxx::normal_iterator< _Ite, _Cont > __first, ::gnu_cxx::normal_<-  
iterator< _Ite, _Cont > __last, const _Tp &__value)`
- `constexpr void __fill_a1 (::Bit_iterator, ::Bit_iterator, const bool &)`
- `template<typename _ForwardIterator, typename _Tp>  
constexpr void __fill_a1 (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Up, typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||__<-  
memcpyable_integer< _Tp >::__width), void >::__type __fill_a1 (_Up *__first, _Up *__last, const _Tp &__x)`
- `template<typename _Tp, typename _VTp>  
void __fill_a1 (const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque_iterator< _Tp, _Tp &,  
_Tp * > &__last, const _VTp &__value)`
- `constexpr void __fill_bvector (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept`
- `constexpr void __fill_bvector_n (_Bit_type *, size_t, bool) noexcept`
- `template<typename _OutputIterator, typename _Size, typename _Tp>  
constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>  
constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>  
constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>  
constexpr ::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > __fill_n_a (const ::gnu_debug::Safe_iterator<  
_Ite, _Seq, _Cat > & __first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>  
constexpr _OutputIterator __fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate>  
constexpr _BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, <-  
_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag,  
_BinaryPredicate __comp)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _Iterator, typename _Predicate>`  
`constexpr _Iterator find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance>`  
`constexpr _InputIterator find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _Tp, typename... _Types>`  
`constexpr size_t find_uniq_type_in_pack ()`
- `from_chars_result from_chars_bfloat16_t (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars_format::general) noexcept`
- `from_chars_result from_chars_float16_t (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars_format::general) noexcept`
- `template<typename _EuclideanRingElement>`  
`constexpr _EuclideanRingElement gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _UInt1, typename _UInt2, typename _UComm = _conditional_t<(sizeof(_UInt2) > sizeof(_UInt1)), _UInt2, _UInt1>>`  
`constexpr __gen_canon_log_res< _UInt1 > gen_canon_log (_UInt1 __val, _UInt2 __base)`
- `template<typename _IntType, typename _UniformRandomBitGenerator>`  
`pair< _IntType, _IntType > gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator &&__g)`
- `template<typename _RealT, size_t __d, typename _Urbg>`  
`_RealT generate_canonical_any (_Urbg &__urng)`
- `template<typename _RealT, size_t __d, typename _Urbg>`  
`_RealT generate_canonical_pow2 (_Urbg &__urng)`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & get_helper (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & get_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<size_t __i, typename... _Types>`  
`__enable_if_t<(__i >= sizeof...( _Types))> get_helper (const tuple< _Types... > &__t)=delete`
- `void glibcxx_assert_fail (const char *__file, int __line, const char *__function, const char *__condition) noexcept`
- `template<typename _Tp>`  
`size_t iconv_adapter (size_t>(* __func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char ** __inbuf, size_t * __inbytes, char ** __outbuf, size_t * __outbytes)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT>`  
`int int_to_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios\_base::fmtflags __flags, bool __dec)`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && invfwd (typename remove\_reference< _Tp >::type &__t) noexcept`

- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result< _Callable, _Args... >::type __invoke ( _Callable &&__fn, _Args &&... __args)`  
`noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res __invoke_impl ( __invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res __invoke_impl ( __invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > __invoke_r ( _Callable &&__fn,`  
`_Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)`
- `constexpr bool __is_constant_evaluated () noexcept`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance>`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare>`  
`constexpr _Distance __is_heap_until ( _RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool __is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool __is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _Rng>`  
`constexpr bool __is_power_of_2_less_1 ( _Rng __range)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __is_sorted_until ( _ForwardIterator __first, _ForwardIterator __last, _Compare &__comp)`
- `template<typename _CharT, typename _Traits>`  
`void __istream_extract (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)`
- `void __istream_extract (istream &, char *, streamsize)`
- `template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>`  
`int __lex_cmp_dit ( ::Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::Deque_iterator< _Tp1, _Ref, _Ptr > __last1,`  
`const _Tp2 * __first2, const _Tp2 * __last2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool __lexicographical_compare_aux ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux ( _II1 __first1, _II1 __last1, const ::gnu_debug::Safe_iterator<`  
`_Ite2, _Seq2, _Cat2 > & __first2, const ::gnu_debug::Safe_iterator< _Ite2, _Seq2, _Cat2 > & __last2)`
- `template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux ( _II1 __first1, _II1 __last1, const ::gnu_debug::Safe_iterator<`  
`_Ite2, _Seq2, _Cat2 > & __first2, const ::gnu_debug::Safe_iterator< _Ite2, _Seq2, _Cat2 > & __last2)`

- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _II2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1`  
`> &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _II2 __first2, _II2 __last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, ↵`  
`_Cat1 > &__first1, const ::__gnu_debug::__Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const`  
`::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< ↵`  
`_Ite2, _Seq2, _Cat2 > &__last2)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _II2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1`  
`> &, const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _II2, _II2)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>`  
`constexpr bool __lexicographical_compare_aux (const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1`  
`> &, const ::__gnu_debug::__Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator<`  
`_Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::__Safe_iterator< _Iter2, _Seq2, _Cat2 > &)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool __lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator<`  
`_Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator< _Tp2,`  
`_Ref2, _Ptr2 > __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool __lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator<`  
`_Tp1, _Ref1, _Ptr1 > __last1, _Tp2 * __first2, _Tp2 * __last2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool __lexicographical_compare_aux1 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool __lexicographical_compare_aux1 (_Tp1 * __first1, _Tp1 * __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2`  
`> __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _II1, typename _II2, typename _Compare>`  
`constexpr bool __lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, ↵`  
`_Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp __lg (_Tp __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator __lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,`  
`_Compare __comp)`
- `template<typename... _BoundArgs, typename... _Args>`  
`constexpr auto __make_bound_args (_Args &&... __args)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &↵`  
`__comp)`
- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<↵`  
`_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move ↵`  
`iterator<_Tp*>>>>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Tp * __i)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator< _Iterator > __make_reverse_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > __make_shared (_Args &&... __args)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare ↵`  
`comp)`



- `template<typename _Tp, typename _Up>`  
`constexpr int __memcmp (const _Tp * __first1, const _Up * __first2, size_t __num)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void __merge_adaptive_resize (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>`  
`void __merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare>`  
`constexpr void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr auto __min_cmp (_Tp __x, _Tp __y)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator __min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair<_ForwardIterator, _ForwardIterator> __minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> __mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> __mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator>`  
`constexpr _Iterator __miter_base (_Iterator __it)`
- `template<typename _Iterator, typename _Compare>`  
`constexpr void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>`  
`void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool __next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`

- `template<typename _Ite, typename _Seq>`  
`constexpr decltype(std::__niter_base(std::declval<_Ite>())) __niter_base (const ::__gnu_debug::__Safe_iterator<`  
`_Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible<_Ite>::↵`  
`value)`
- `void __once_proxy (void)`
- `template<typename _BidirectionalIterator, typename _Predicate>`  
`constexpr _BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate`  
`__pred, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred,`  
`forward_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Random↵`  
`AccessIterator __result, _Compare &__comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool __prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __↵`  
`comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
`constexpr void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex,`  
`_Tp __value, _Compare &__comp)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter __regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex<_Ch_type, ↵`  
`Rx_traits > &__e, const _Ch_type * __fmt, size_t __len, regex_constants::match_flag_type __flags)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator __remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator __remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator __replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, ↵`  
`_BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __↵`  
`last, forward_iterator_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle,`  
`_RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>`  
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵`  
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance ↵`  
`__buffer_size)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵`  
`Generator>`  
`_OutputIterator __sample (_ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _OutputIterator`  
`__out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)`



- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>`  
`_RandomAccessIterator __sample (_InputIterator __first, _InputIterator __last, input\_iterator\_tag, _RandomAccessIterator __out, random\_access\_iterator\_tag, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 __search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator __search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>`  
`constexpr _RandomAccessIter __search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator __set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`_Tp * __shared_ptr_deref (_Tp * __p)`
- `constexpr long long __size_to_integer (double __n)`
- `constexpr long long __size_to_integer (float __n)`
- `constexpr int __size_to_integer (int __n)`
- `constexpr long __size_to_integer (long __n)`
- `constexpr long long __size_to_integer (long double __n)`
- `constexpr long long __size_to_integer (long long __n)`
- `constexpr unsigned __size_to_integer (unsigned __n)`
- `constexpr unsigned long __size_to_integer (unsigned long __n)`
- `constexpr unsigned long long __size_to_integer (unsigned long long __n)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare & __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator __stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>`  
`constexpr _ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void __stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>`  
`void __stable_sort_adaptive_resize (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _Str>`  
`constexpr _Str __str_concat (typename _Str::value_type const * __lhs, typename _Str::size_type __lhs_len, typename _Str::value_type const * __rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type const & __a)`
- `constexpr size_t __sv_check (size_t __size, size_t __pos, const char * __s)`
- `constexpr size_t __sv_limit (size_t __size, size_t __pos, size_t __off) noexcept`
- `void __terminate () noexcept`
- `void __throw_bad_alloc (void)`
- `void __throw_bad_array_new_length (void)`
- `void __throw_bad_cast (void)`
- `void __throw_bad_exception (void)`
- `void __throw_bad_function_call ()`
- `void __throw_bad_typeid (void)`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *)`
- `void __throw_future_error (int)`
- `void __throw_invalid_argument (const char *)`
- `void __throw_ios_failure (const char *)`
- `void __throw_ios_failure (const char *, int)`
- `void __throw_length_error (const char *)`
- `void __throw_logic_error (const char *)`
- `void __throw_out_of_range (const char *)`
- `void __throw_out_of_range_fmt (const char *,...)`
- `void __throw_overflow_error (const char *)`
- `void __throw_range_error (const char *)`
- `void __throw_runtime_error (const char *)`
- `void __throw_system_error (int)`
- `void __throw_underflow_error (const char *)`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_i (char * __first, char * __last, _Tp __value, int __base=10)`
- `constexpr void __to_wstring_numeric (const char * __s, int __len, wchar_t * __wout)`
- `wstring __to_wstring_numeric (const string & __s)`
- `template<typename _Facet>`  
`const _Facet * __try_use_facet (const locale & __loc)`
- `template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>`  
`constexpr _Cat __tuple_cmp (const _Tp & __t, const _Up & __u, _IndexSeq __indices)`
- `template<typename _Tp, typename _ForwardIterator>`  
`void __uninitialized_construct_buf (_Tp * __first, _Tp * __last, _ForwardIterator __seed)`
- `template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > __uninitialized_copy_a (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)`
- `template<typename _Iter, typename _OTp, typename _Tp>`  
`__enable_if_t< __is_random_access_iter< _Iter >::value, ::Deque_iterator< _OTp, _OTp &, _OTp * > > < __uninitialized_copy_a (_Iter __first, _Iter __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)`
- `template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > __uninitialized_move_a (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)`

- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator __unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __↵`  
`_result, _BinaryPredicate __binary_pred, forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred, input\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`_ForwardIterator __unique_copy_1 (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, __↵`  
`_BinaryPredicate __binary_pred, __true_type)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator __unique_copy_1 (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _BinaryPredicate __binary_pred, __false_type)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator __upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,`  
`_Compare __comp)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __k)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array<`  
`size_t > __i)`
- `template<typename _Tp>`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array<`  
`size_t > __j)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t >`  
`__i)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`

- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp>`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp>`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`

- `template<typename _Tp>`  
`_Tp * __valarray_get_storage (size_t)`
- `template<typename _Ta>`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta>`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp>`  
`void __valarray_release_memory (_Tp *__p, size_t __n)`
- `template<typename _Tp>`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<template< typename > class _Trait, typename _Tp, typename _Up = _Tp>`  
`constexpr _Up __value_or (_Up __def=_Up()) noexcept`
- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<typename _CharT, typename _OutIter>`  
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _CharT>`  
`ostreambuf\_iterator< _CharT > __write (ostreambuf\_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`

- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__minus (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__minus (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, _Array<size_t> __i, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__modulus (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array<_Tp> __a, _Array<bool> __m, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__multiplies (_Array<_Tp> __a, _Array<bool> __m, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`



- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`  
`__n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`

- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp>`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp>`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, typename... _Args>`  
`constexpr void _Construct (_Tp *__p, _Args &&... __args)`
- `template<typename _T1>`  
`constexpr void _Construct_novalue (_T1 *__p)`
- `template<typename _ForwardIterator>`  
`constexpr void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp>`  
`constexpr void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Tp abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > abs (const valarray< _Tp > &__v)`
- `constexpr double abs (double __x)`
- `constexpr float abs (float __x)`
- `long abs (long __i)`
- `constexpr long double abs (long double __x)`
- `long long abs (long long __x)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type acos (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`std::complex< _Tp > acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > acos (const valarray< _Tp > &__v)`

- constexpr float **acos** (float \_\_x)
- constexpr long double **acos** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **acosh** (\_Tp \_\_x)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **acosh** (const [std::complex](#)< \_Tp > &)
- constexpr float **acosh** (float \_\_x)
- constexpr long double **acosh** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_Tp \* **addressof** (\_Tp &\_\_r) noexcept
- template<typename \_Tp>  
const \_Tp \* **addressof** (const \_Tp &&)=delete
- template<typename \_InputIterator, typename \_OutputIterator>  
constexpr \_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_↵  
result)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_BinaryOperation>  
constexpr \_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_↵  
result, \_BinaryOperation \_\_binary\_op)
- template<typename \_Filter>  
constexpr \_Filter **adjacent\_find** (\_Filter, \_Filter)
- template<typename \_Filter, typename \_BinaryPredicate>  
constexpr \_Filter **adjacent\_find** (\_Filter, \_Filter, \_BinaryPredicate)
- template<typename \_ForwardIterator>  
constexpr \_ForwardIterator **adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator, typename \_BinaryPredicate>  
constexpr \_ForwardIterator **adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate ↵  
\_\_binary\_pred)
- template<typename \_InputIterator, typename \_Distance>  
constexpr void **advance** (\_InputIterator &\_\_i, \_Distance \_\_n)
- template<typename \_CharT, typename \_Distance>  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, void >::\_\_type **advance** ([istreambuf\\_iterator](#)< ↵  
CharT > &\_\_i, \_Distance \_\_n)
- void \* **align** (size\_t \_\_align, size\_t \_\_size, void \*&\_\_ptr, size\_t &\_\_space) noexcept
- template<typename \_Iter, typename \_Predicate>  
constexpr bool **all\_of** (\_Iter, \_Iter, \_Predicate)
- template<typename \_InputIterator, typename \_Predicate>  
constexpr bool **all\_of** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_Iter, typename \_Predicate>  
constexpr bool **any\_of** (\_Iter, \_Iter, \_Predicate)
- template<typename \_InputIterator, typename \_Predicate>  
constexpr bool **any\_of** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **arg** (\_Tp \_\_x)
- template<typename \_Tp>  
\_Tp **arg** (const [complex](#)< \_Tp > &)
- template<typename \_Tp, typename... \_Up>  
**array** (\_Tp, \_Up...) -> array< [enable\\_if\\_t](#)<(is\_same\_v< \_Tp, \_Up > &&...), \_Tp >, 1+sizeof...(Up)>
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **asin** (\_Tp \_\_x)
- template<class \_Dom>  
\_Expr< \_UnClos< struct std::\_Asin, \_Expr, \_Dom >, typename \_Dom::value\_type > **asin** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e)

- `template<typename _Tp>`  
`std::complex< _Tp > asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &_v)`
- `constexpr float asin (float __x)`
- `constexpr long double asin (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asinh (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > asinh (const std::complex< _Tp > &)`
- `constexpr float asinh (float __x)`
- `constexpr long double asinh (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<size_t _Align, class _Tp>`  
`constexpr _Tp * assume_aligned (_Tp * __ptr) noexcept`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (_Fn && __fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch __policy, _Fn && __fn, _Args &&... __args)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type atan (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`std::complex< _Tp > atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &_v)`
- `constexpr float atan (float __x)`
- `constexpr long double atan (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std:: _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std:: _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2 (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std:: _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std:: _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std:: _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float atan2 (float __y, float __x)`
- `constexpr long double atan2 (long double __y, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type atanh (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &)`
- `constexpr float atanh (float __x)`
- `constexpr long double atanh (long double __x)`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`

- `template<typename _ITp>`  
`_ITp atomic_exchange_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp>`  
`_ITp atomic_exchange_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_add_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order`  
`__m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order`  
`__m) noexcept`

- `template<typename _ITp>`  
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_flag_clear (atomic_flag *__a) noexcept`
- `void atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`bool atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`bool atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `void atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_store (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void atomic_store (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void atomic_store_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void atomic_store_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Container>`  
`constexpr back_insert_iterator< _Container > back_inserter (_Container &__x)`
- `template<typename _ForwardIterator>`  
`basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type=) -> basic_regex< typename iterator_traits< _ForwardIterator >::value_type >`
- `template<typename _InputIterator, typename _CharT = typename iterator_traits< _InputIterator >::value_type, typename _Allocator = allocator< _CharT >, typename = _RequireInputIter< _InputIterator >, typename = _RequireAllocator< _Allocator >>`  
`basic_string (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >`
- `template<typename _CharT, typename _Traits, typename _Allocator = allocator< _CharT >, typename = _RequireAllocator< _Allocator >>`  
`basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >`



- `template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>>`  
**basic\_string** ([basic\\_string\\_view](#)< \_CharT, \_Traits >, typename `basic_string`< \_CharT, \_Traits, \_Allocator >::size\_type, typename `basic_string`< \_CharT, \_Traits, \_Allocator >::size\_type, const \_Allocator &= \_Allocator())  
 -> `basic_string`< \_CharT, \_Traits, \_Allocator >
- `template<typename _Operation>`  
 void **basic\_string**< \_CharT, \_Traits, \_Alloc > **resize\_and\_overwrite** (const size\_type \_\_n, \_Operation \_\_op)
- `template<contiguous_iterator _It, sized_sentinel_for< _It > _End>`  
**basic\_string\_view** ( \_It, \_End) -> `basic_string_view`< iter\_value\_t< \_It > >
- `template<ranges::contiguous_range _Range>`  
**basic\_string\_view** ( \_Range &&) -> `basic_string_view`< ranges::range\_value\_t< \_Range > >
- `template<typename _Container>`  
 constexpr auto [begin](#) ( \_Container &\_\_cont) noexcept(noexcept(\_\_cont.begin())) -> decltype(\_\_cont.begin())
- `template<typename _Tp, size_t _Nm>`  
 constexpr \_Tp \* [begin](#) ( \_Tp(&\_\_arr)[\_Nm]) noexcept
- `template<typename _Container>`  
 constexpr auto [begin](#) (const \_Container &\_\_cont) noexcept(noexcept(\_\_cont.begin())) -> decltype(\_\_cont.begin())
- `template<typename _Tpa, typename _Tpb>`  
 \_\_gnu\_cxx::\_\_promote\_2< \_Tpa, \_Tpb >::\_\_type [beta](#) ( \_Tpa \_\_a, \_Tpb \_\_b)
- float [betaf](#) (float \_\_a, float \_\_b)
- long double [betal](#) (long double \_\_a, long double \_\_b)
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
 constexpr bool **binary\_search** ( \_Filter, \_Filter, const \_Tp &)
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
 constexpr bool **binary\_search** ( \_Filter, \_Filter, const \_Tp &, \_Compare)
- `template<typename _ForwardIterator, typename _Tp>`  
 constexpr bool [binary\\_search](#) ( \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val)
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
 constexpr bool [binary\\_search](#) ( \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
 constexpr \_Bindres\_helper< \_Result, \_Func, \_BoundArgs... >::type [bind](#) ( \_Func &&\_\_f, \_BoundArgs &&... \_\_args)
- `template<typename _Func, typename... _BoundArgs>`  
 constexpr \_Bind\_helper< \_\_is\_socketlike< \_Func >::value, \_Func, \_BoundArgs... >::type [bind](#) ( \_Func &&\_\_f, \_BoundArgs &&... \_\_args)
- `template<typename _Operation, typename _Tp>`  
[binder1st](#)< \_Operation > [bind1st](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)
- `template<typename _Operation, typename _Tp>`  
[binder2nd](#)< \_Operation > [bind2nd](#) (const \_Operation &\_\_fn, const \_Tp &\_\_x)
- `ios_base & boolalpha` (`ios_base` &\_\_base)
- `template<typename _Callable, typename... _Args>`  
 void [call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)
- `template<typename _IndexType, size_t... _Extents, typename... _RawSlices>`  
 constexpr auto **canonical\_slices** (const extents< \_IndexType, \_Extents... > &\_\_exts, \_RawSlices... \_\_raw\_slices)
- `template<typename _Container>`  
 constexpr auto [cbegin](#) (const \_Container &\_\_cont) noexcept(noexcept(std::begin(\_\_cont))) -> decltype(std::begin(\_\_cont))
- `template<typename _Tp>`  
 constexpr \_\_gnu\_cxx::enable\_if< \_\_is\_integer< \_Tp >::value, double >::\_\_type **cbrt** ( \_Tp \_\_x)
- `constexpr float cbrt` (float \_\_x)

- constexpr long double **cbrt** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **ceil** (\_Tp \_\_x)
- constexpr float **ceil** (float \_\_x)
- constexpr long double **ceil** (long double \_\_x)
- template<typename \_Container>  
constexpr auto **cend** (const \_Container &\_\_cont) noexcept(noexcept(std::end(\_\_cont))) -> decltype(std::end(\_←  
\_\_cont))
- template<typename \_Tp>  
constexpr const \_Tp & **clamp** (const \_Tp &\_\_val, const \_Tp &\_\_lo, const \_Tp &\_\_hi)
- template<typename \_Tp, typename \_Compare>  
constexpr const \_Tp & **clamp** (const \_Tp &\_\_val, const \_Tp &\_\_lo, const \_Tp &\_\_hi, \_Compare \_\_comp)
- template<typename \_Tp, typename \_Ap>  
constexpr basic\_vec< \_Tp, \_Ap > **clamp** (const basic\_vec< \_Tp, \_Ap > &\_\_v, const basic\_vec< \_Tp, \_Ap >  
&\_\_lo, const basic\_vec< \_Tp, \_Ap > &\_\_hi)
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_equal** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_greater** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_greater\_equal** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_less** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_less\_equal** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr bool **cmp\_not\_equal** (\_Tp \_\_t, \_Up \_\_u) noexcept
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **comp\_ellint\_1** (\_Tp \_\_k)
- float **comp\_ellint\_1f** (float \_\_k)
- long double **comp\_ellint\_1l** (long double \_\_k)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **comp\_ellint\_2** (\_Tp \_\_k)
- float **comp\_ellint\_2f** (float \_\_k)
- long double **comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tp>  
constexpr std::complex< typename \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type > **conj** (\_Tp \_\_x)
- template<typename \_Tp>  
constexpr complex< \_Tp > **conj** (const complex< \_Tp > &)
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > **const\_pointer\_cast** (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_r) noexcept
- template<typename \_Tp, typename... \_Args>  
constexpr \_Tp \* **construct\_at** (\_Tp \*\_\_location, \_Args &&... \_\_args) noexcept(noexcept(::new((void \*) 0) \_←  
Tp(std::declval< \_Args >()...)))
- template<typename \_II, typename \_OI>  
constexpr \_OI **copy** (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- template<typename \_IIter, typename \_OIter>  
constexpr \_OIter **copy** (\_IIter, \_IIter, \_OIter)

- `template<typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT >::__type copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT`  
`> __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Biter1, typename _Biter2>`  
`constexpr _Biter2 copy_backward (_Biter1, _Biter1, _Biter2)`
- `template<typename _Iiter, typename _Oiter, typename _Predicate>`  
`constexpr _Oiter copy_if (_Iiter, _Iiter, _Oiter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate`  
`__pred)`
- `template<typename _Iiter, typename _Size, typename _Oiter>`  
`constexpr _Oiter copy_n (_Iiter, _Size, _Oiter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type copysign (_Tp __x, _Up __y)`
- `constexpr float copysign (float __x, float __y)`
- `constexpr long double copysign (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cos (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Cos, _Expr, _Dom >, typename _Dom::value_type > cos (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > cos (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &__v)`
- `constexpr float cos (float __x)`
- `constexpr long double cos (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cosh (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Cosh, _Expr, _Dom >, typename _Dom::value_type > cosh (const _Expr< _↵`  
`Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > cosh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &__v)`
- `constexpr float cosh (float __x)`
- `constexpr long double cosh (long double __x)`
- `template<typename _Iiter, typename _Tp = typename iterator_traits< _Iiter >::value_type>`  
`constexpr iterator_traits< _Iiter >::difference_type count (_Iiter, _Iiter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last,`  
`const _Tp &__value)`
- `template<typename _Iiter, typename _Predicate>`  
`constexpr iterator_traits< _Iiter >::difference_type count_if (_Iiter, _Iiter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last,`  
`_Predicate __pred)`

- `template<typename _Container>`  
`constexpr auto crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↵  
__cont))`
- `template<typename _Container>`  
`constexpr auto crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↵  
__cont))`
- `exception_ptr current_exception ()` `noexcept`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Container>`  
`constexpr auto data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * data (_Tp(&__array)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `ios_base & dec (ios_base &__base)`
- `template<typename _Tp>`  
`auto declval () noexcept -> decltype(__declval< _Tp >())`
- `ios_base & defaultfloat (ios_base &__base)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator =`  
`allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >`
- `template<typename _ForwardIterator>`  
`constexpr void destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp>`  
`constexpr void destroy_at (_Tp *__location)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator destroy_n (_ForwardIterator __first, _Size __count)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::difference_type distance (_InputIterator __first, _InputIterator __last)`
- `ldiv_t div (long __i, long __j) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`

- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 ( _Tp __k, _Tpp __phi)`
- `float ellint_2f (float __k, float __phi)`
- `long double ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float ellint_3f (float __k, float __nu, float __phi)`
- `long double ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Container>`  
`constexpr auto empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__cont.empty())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool empty (const _Tp(&)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto end ( _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * end ( _Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _II1, typename _II2>`  
`constexpr bool equal ( _II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2>`  
`constexpr bool equal ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>`  
`constexpr bool equal ( _IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>`  
`constexpr bool equal ( _IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2>`  
`constexpr bool equal ( _IIter1, _IIter1, _IIter2)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
`constexpr pair< _Filter, _Filter > equal_range ( _Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, size_t _Nm, typename _Up = _Tp>`  
`constexpr size_t erase (::inplace_vector< _Tp, _Nm > &__cont, const _Up &__value)`
- `template<typename _Tp, size_t _Nm, typename _Predicate>`  
`constexpr size_t erase_if (::inplace_vector< _Tp, _Nm > &__cont, _Predicate __pred)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::__value, double >::__type erf ( _Tp __x)`
- `constexpr float erf (float __x)`
- `constexpr long double erf (long double __x)`

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type erfc (_Tp __x)`
- `constexpr float erfc (float __x)`
- `constexpr long double erfc (long double __x)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `constexpr float exp (float __x)`
- `constexpr long double exp (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp2 (_Tp __x)`
- `constexpr float exp2 (float __x)`
- `constexpr long double exp2 (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type expm1 (_Tp __x)`
- `constexpr float expm1 (float __x)`
- `constexpr long double expm1 (long double __x)`
- `template<typename... _Integrals>`  
`extents (_Integrals...) -> extents< size_t, __detail::__maybe_static_ext< _Integrals >... >`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type fabs (_Tp __x)`
- `template<typename _Tp>`  
`_Tp fabs (const std::complex< _Tp > &__z)`
- `constexpr float fabs (float __x)`
- `constexpr long double fabs (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fdim (_Tp __x, _Up __y)`
- `constexpr float fdim (float __x, float __y)`
- `constexpr long double fdim (long double __x, long double __y)`
- `template<typename _Filter, typename _Tp = typename iterator_traits< _Filter >::value_type>`  
`constexpr void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp>`  
`constexpr _OI fill_n (_OI __first, _Size __n, const _Tp &__value)`

- `template<typename _Olter, typename _Size, typename _Tp = typename iterator_traits<_Olter>::value_type>  
constexpr _Olter fill_n (_Olter, _Size, const _Tp &)`
- `template<typename _Ilter, typename _Tp = typename iterator_traits<_Ilter>::value_type>  
constexpr _Ilter find (_Ilter, _Ilter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp>  
constexpr _InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp & __val)`
- `template<typename _CharT>  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT >::__type find  
(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _Filter1, typename _Filter2>  
constexpr _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>  
constexpr _Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>  
constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2  
__first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>  
constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2  
__first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2>  
constexpr _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>  
constexpr _Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator>  
constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
_ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>  
constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵  
_ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _Ilter, typename _Predicate>  
constexpr _Ilter find_if (_Ilter, _Ilter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>  
constexpr _InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Ilter, typename _Predicate>  
constexpr _Ilter find_if_not (_Ilter, _Ilter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>  
constexpr _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & fixed (ios_base & __base)`
- `template<typename _Tp>  
constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type floor (_Tp __x)`
- `constexpr float floor (float __x)`
- `constexpr long double floor (long double __x)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & flush (basic\_ostream< _CharT, _Traits > & __os)`
- `template<typename _Tp, typename _Up, typename _Vp>  
constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type fma (_Tp __x, _Up __y, _Vp __z)`
- `constexpr float fma (float __x, float __y, float __z)`
- `constexpr long double fma (long double __x, long double __y, long double __z)`
- `template<typename _Tp, typename _Up>  
constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type fmax (_Tp __x, _Up __y)`
- `constexpr float fmax (float __x, float __y)`
- `constexpr long double fmax (long double __x, long double __y)`

- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2<_Tp, _Up>::__type fmin (_Tp __x, _Up __y)`
- `constexpr float fmin (float __x, float __y)`
- `constexpr long double fmin (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2<_Tp, _Up>::__type fmod (_Tp __x, _Up __y)`
- `constexpr float fmod (float __x, float __y)`
- `constexpr long double fmod (long double __x, long double __y)`
- `template<typename _Iter, typename _Funct>`  
`constexpr _Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function>`  
`constexpr _Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`  
`constexpr _InputIterator for_each_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _Tp>`  
`constexpr _Tp && forward (typename std::remove\_reference<_Tp>::__type &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp && forward (typename std::remove\_reference<_Tp>::__type &__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple<_Elements &&...> forward_as_tuple (_Elements &&... __args) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr __like_t<_Tp, _Up> forward_like (_Up &&__x) noexcept`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward\_list<_ValT, _Allocator>`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if<__is_integer<_Tp>::__value, int>::__type fpclassify (_Tp __x)`
- `constexpr int fpclassify (double __x)`
- `constexpr int fpclassify (float __x)`
- `constexpr int fpclassify (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type frexp (_Tp __x, int *__exp)`
- `float frexp (float __x, int *__exp)`
- `long double frexp (long double __x, int *__exp)`
- `template<typename _Tp, enable\_if\_t<__or<__is_signed_or_unsigned_integer<_Tp>, is_same<char, _Tp>::__value, int> = 0>`  
`constexpr from\_chars\_result from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)`
- `from\_chars\_result from_chars (const char *__first, const char *__last, double &__value, chars\_format __fmt=chars_format::general) noexcept`
- `from\_chars\_result from_chars (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars_format::general) noexcept`
- `from\_chars\_result from_chars (const char *__first, const char *__last, long double &__value, chars\_format __fmt=chars_format::general) noexcept`
- `template<typename _Container>`  
`constexpr front\_insert\_iterator<_Container> front_inserter (_Container &__x)`
- `template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>`  
`function (_Fn) -> function<_Signature>`
- `template<typename _Res, typename... _ArgTypes>`  
`function (_Res(*)(_ArgTypes...)) -> function<_Res(_ArgTypes...)>`
- `template<typename _Fn>`  
`function_ref (_Fn *) -> function_ref<_Fn>`



- `template<auto __fn, typename _Fn>`  
**function\_ref** (constant\_wrapper< \_\_fn, \_Fn > -> function\_ref< [remove\\_pointer\\_t](#)< \_Fn > >
- `template<auto __fn, typename _Fn, typename _Tp, typename _SignaturePtr = decltype(__polyfunc::__deduce_funcref<_Fn, _Tp&>())>`  
**function\_ref** (constant\_wrapper< \_\_fn, \_Fn >, \_Tp && -> function\_ref< [remove\\_pointer\\_t](#)< \_SignaturePtr >  
 >
- `const error\_category & future\_category ()` noexcept
- `template<typename _Filter, typename _Generator>`  
 constexpr void **generate** (\_Filter, \_Filter, \_Generator)
- `template<typename _ForwardIterator, typename _Generator>`  
 constexpr void [generate](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Generator \_\_gen)
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>`  
 \_RealType [generate\\_canonical](#) (\_UniformRandomNumberGenerator &\_\_g)
- `template<typename _RealT, size_t __digits, typename _Urbg>`  
 \_RealT [generate\\_canonical](#) (\_Urbg &\_\_urng)
- `template<typename _OIter, typename _Size, typename _Generator>`  
 constexpr \_OIter **generate\_n** (\_OIter, \_Size, \_Generator)
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
 constexpr \_OutputIterator [generate\\_n](#) (\_OutputIterator \_\_first, \_Size \_\_n, \_Generator \_\_gen)
- `const error\_category & generic\_category ()` noexcept
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
 constexpr \_Tp && **get** (array< \_Tp, \_Nm > &&\_\_arr) noexcept
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
 constexpr \_Tp & **get** (array< \_Tp, \_Nm > &\_\_arr) noexcept
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
 constexpr const \_Tp && **get** (const array< \_Tp, \_Nm > &&\_\_arr) noexcept
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
 constexpr const \_Tp & **get** (const array< \_Tp, \_Nm > &\_\_arr) noexcept
- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
 constexpr auto **get** (const subrange< \_It, \_Sent, \_Kind > &\_\_r)
- `template<size_t __i, typename... _Elements>`  
 constexpr const \_\_tuple\_element\_t< \_\_i, tuple< \_Elements... > > && [get](#) (const tuple< \_Elements... > &&\_\_t)  
 noexcept
- `template<size_t __i, typename... _Elements>`  
 constexpr const \_\_tuple\_element\_t< \_\_i, tuple< \_Elements... > > & [get](#) (const tuple< \_Elements... > &\_\_t)  
 noexcept
- `template<size_t __i, typename... _Elements>`  
 constexpr \_\_tuple\_element\_t< \_\_i, tuple< \_Elements... > > && [get](#) (tuple< \_Elements... > &&\_\_t) noexcept
- `template<size_t __i, typename... _Elements>`  
 constexpr \_\_tuple\_element\_t< \_\_i, tuple< \_Elements... > > & [get](#) (tuple< \_Elements... > &\_\_t) noexcept
- **Catalogs & [get\\_catalogs](#) ()**
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
 \_Del \* **get\_deleter** (const \_\_shared\_ptr< \_Tp, \_Lp > &\_\_p) noexcept
- `template<typename _Del, typename _Tp>`  
 \_Del \* [get\\_deleter](#) (const [shared\\_ptr](#)< \_Tp > &\_\_p) noexcept
- `template<typename _MoneyT>`  
 \_Get\_money< \_MoneyT > [get\\_money](#) (\_MoneyT &\_\_mon, bool \_\_intl=false)
- `new\_handler get\_new\_handler ()` noexcept
- `template<typename _Tp>`  
 pair< \_Tp \*, ptrdiff\_t > [get\\_temporary\\_buffer](#) (ptrdiff\_t \_\_len) noexcept
- `terminate\_handler get\_terminate ()` noexcept
- `template<typename _CharT>`  
 \_Get\_time< \_CharT > [get\\_time](#) (std::tm \*\_\_tmb, const \_CharT \*\_\_fmt)

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > && __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > && __is, basic_string< _CharT, _Traits, _Alloc > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Alloc > & __str, _CharT __delim)`
- `template<> basic_istream< char > & getline (basic_istream< char > & __in, basic_string< char > & __str, char __delim)`
- `template<> basic_istream< wchar_t > & getline (basic_istream< wchar_t > & __in, basic_string< wchar_t > & __str, wchar_t __delim)`
- `template<typename _Facet>`  
`bool has_facet (const locale & __loc)`
- `template bool has_facet< codecvt< char, char, mbstate_t > > (const locale &)`
- `template bool has_facet< codecvt< wchar_t, char, mbstate_t > > (const locale &)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `ios_base & hex (ios_base & __base)`
- `ios_base & hexfloat (ios_base & __base)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type hypot (_Tp __x, _Up __y)`
- `constexpr float hypot (float __x, float __y)`
- `constexpr long double hypot (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type ilogb (_Tp __x)`
- `constexpr int ilogb (float __x)`
- `constexpr int ilogb (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type imag (_Tp)`
- `template<typename _Tp>`  
`constexpr _Tp imag (const complex< _Tp > & __z)`
- `template<typename _Res, typename _Tp>`  
`constexpr bool in_range (_Tp __t) noexcept`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare>`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`  
`constexpr _OutputIterator inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _Pointer = void, typename _Smart, typename... _Args>`  
`auto inout\_ptr (_Smart &__s, _Args &&... __args)`
- `template<typename _BidirectionalIterator>`  
`constexpr void inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr void inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BIter>`  
`constexpr void inplace\_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr void inplace\_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _Container>`  
`constexpr insert\_iterator<_Container> insertiter (_Container &__x, std::__detail::__range_iter_t<_Container> __i)`
- `ios\_base & internal (ios\_base &__base)`
- `const error\_category & iostream\_category () noexcept`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `constexpr bool is\_eq (partial_ordering __cmp) noexcept`
- `constexpr bool is\_gt (partial_ordering __cmp) noexcept`
- `constexpr bool is\_gteq (partial_ordering __cmp) noexcept`
- `template<typename _RAIter>`  
`constexpr bool is\_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr bool is\_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter>`  
`constexpr _RAIter is\_heap\_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr _RAIter is\_heap\_until (_RAIter, _RAIter, _Compare)`

- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __↵`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator is\_heap\_until (_RandomAccessIterator __first, _RandomAccessIterator __↵`  
`last, _Compare __comp)`
- `constexpr bool is\_lt (partial_ordering __cmp) noexcept`
- `constexpr bool is\_lteq (partial_ordering __cmp) noexcept`
- `constexpr bool is\_neq (partial_ordering __cmp) noexcept`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool is\_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr bool is\_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr bool is\_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`  
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _Filter>`  
`constexpr bool is\_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr bool is\_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr bool is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter>`  
`constexpr _Filter is\_sorted\_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter is\_sorted\_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<size_t _Align, class _Tp>`  
`bool is\_sufficiently\_aligned (_Tp *__ptr)`
- `template<typename _CharT>`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isblank (_CharT __c, const locale &__loc)`

- `template<typename _CharT>`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if<__is_integer< _Tp >::__value, bool >::__type isfinite (_Tp)`
- `constexpr bool isfinite (double __x)`
- `constexpr bool isfinite (float __x)`
- `constexpr bool isfinite (long double __x)`
- `template<typename _CharT>`  
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type isgreater (_Tp __x, _Up __y)`
- `constexpr bool isgreater (double __x, double __y)`
- `constexpr bool isgreater (float __x, float __y)`
- `constexpr bool isgreater (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type isgreaterequal (_Tp __x, _Up __y)`
- `constexpr bool isgreaterequal (double __x, double __y)`
- `constexpr bool isgreaterequal (float __x, float __y)`
- `constexpr bool isgreaterequal (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if<__is_integer< _Tp >::__value, bool >::__type isinf (_Tp)`
- `constexpr bool isinf (double __x)`
- `constexpr bool isinf (float __x)`
- `constexpr bool isinf (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type isless (_Tp __x, _Up __y)`
- `constexpr bool isless (double __x, double __y)`
- `constexpr bool isless (float __x, float __y)`
- `constexpr bool isless (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type islessequal (_Tp __x, _Up __y)`
- `constexpr bool islessequal (double __x, double __y)`
- `constexpr bool islessequal (float __x, float __y)`
- `constexpr bool islessequal (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type islessgreater (_Tp __x, _Up __y)`
- `constexpr bool islessgreater (double __x, double __y)`
- `constexpr bool islessgreater (float __x, float __y)`
- `constexpr bool islessgreater (long double __x, long double __y)`
- `template<typename _CharT>`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if<__is_integer< _Tp >::__value, bool >::__type isnan (_Tp)`
- `constexpr bool isnan (double __x)`
- `constexpr bool isnan (float __x)`

- constexpr bool **isnan** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, bool >::\_\_type **isnormal** (\_Tp \_\_x)
- constexpr bool **isnormal** (double \_\_x)
- constexpr bool **isnormal** (float \_\_x)
- constexpr bool **isnormal** (long double \_\_x)
- template<typename \_CharT>  
bool **isprint** (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT>  
bool **ispunct** (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT>  
bool **isspace** (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<(\_\_is\_arithmetic< \_Tp >::\_\_value && \_\_is\_arithmetic< \_Up >::\_\_value), bool >::\_\_type **isunordered** (\_Tp \_\_x, \_Up \_\_y)
- constexpr bool **isunordered** (double \_\_x, double \_\_y)
- constexpr bool **isunordered** (float \_\_x, float \_\_y)
- constexpr bool **isunordered** (long double \_\_x, long double \_\_y)
- template<typename \_CharT>  
bool **isupper** (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_CharT>  
bool **isxdigit** (\_CharT \_\_c, const locale &\_\_loc)
- template<typename \_Filter1, typename \_Filter2>  
constexpr void **iter\_swap** (\_Filter1, \_Filter2)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2>  
constexpr void **iter\_swap** (\_ForwardIterator1 \_\_a, \_ForwardIterator2 \_\_b)
- template<typename \_Tp>  
\_Tp **kill\_dependency** (\_Tp \_\_y) noexcept
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **laguerref** (unsigned int \_\_n, float \_\_x)
- long double **laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **ldexp** (\_Tp \_\_x, int \_\_exp)
- constexpr float **ldexp** (float \_\_x, int \_\_exp)
- constexpr long double **ldexp** (long double \_\_x, int \_\_exp)
- **ios\_base & left** (**ios\_base & \_\_base**)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **legendre** (unsigned int \_\_l, \_Tp \_\_x)
- float **legendref** (unsigned int \_\_l, float \_\_x)
- long double **legendrel** (unsigned int \_\_l, long double \_\_x)
- template<typename \_I1, typename \_I2>  
constexpr bool **lexicographical\_compare** (\_I1 \_\_first1, \_I1 \_\_last1, \_I2 \_\_first2, \_I2 \_\_last2)
- template<typename \_I1, typename \_I2, typename \_Compare>  
constexpr bool **lexicographical\_compare** (\_I1 \_\_first1, \_I1 \_\_last1, \_I2 \_\_first2, \_I2 \_\_last2, \_Compare \_\_comp)
- template<typename \_I1, typename \_I2>  
constexpr bool **lexicographical\_compare** (\_I1, \_I1, \_I2, \_I2)
- template<typename \_I1, typename \_I2, typename \_Compare>  
constexpr bool **lexicographical\_compare** (\_I1, \_I1, \_I2, \_I2, \_Compare)
- template<typename \_InputIter1, typename \_InputIter2>  
constexpr auto **lexicographical\_compare\_three\_way** (\_InputIter1 \_\_first1, \_InputIter1 \_\_last1, \_InputIter2 \_\_first2, \_InputIter2 \_\_last2)

- `template<typename _InputIter1, typename _InputIter2, typename _Comp>`  
`constexpr auto lexicographical\_compare\_three\_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2,`  
`__InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type lgamma (_Tp __x)`
- `constexpr float lgamma (float __x)`
- `constexpr long double lgamma (long double __x)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator =`  
`allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type llrint (_Tp __x)`
- `constexpr long long llrint (float __x)`
- `constexpr long long llrint (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type llround (_Tp __x)`
- `constexpr long long llround (float __x)`
- `constexpr long long llround (long double __x)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > log (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `constexpr float log (float __x)`
- `constexpr long double log (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log10 (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > log10 (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > log10 (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > &__v)`
- `constexpr float log10 (float __x)`
- `constexpr long double log10 (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log1p (_Tp __x)`
- `constexpr float log1p (float __x)`
- `constexpr long double log1p (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log2 (_Tp __x)`
- `constexpr float log2 (float __x)`
- `constexpr long double log2 (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type logb (_Tp __x)`

- constexpr float **logb** (float \_\_x)
- constexpr long double **logb** (long double \_\_x)
- template<typename \_Filter, typename \_Tp = typename iterator\_traits<\_Filter>::value\_type>  
constexpr \_Filter **lower\_bound** (\_Filter, \_Filter, const \_Tp &)
- template<typename \_Filter, typename \_Tp = typename iterator\_traits<\_Filter>::value\_type, typename \_Compare>  
constexpr \_Filter **lower\_bound** (\_Filter, \_Filter, const \_Tp &, \_Compare)
- template<typename \_ForwardIterator, typename \_Tp>  
constexpr \_ForwardIterator **lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare>  
constexpr \_ForwardIterator **lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, ←  
\_Compare \_\_comp)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **lrint** (\_Tp \_\_x)
- constexpr long **lrint** (float \_\_x)
- constexpr long **lrint** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **lround** (\_Tp \_\_x)
- constexpr long **lround** (float \_\_x)
- constexpr long **lround** (long double \_\_x)
- template<input\_iterator \_It>  
constexpr const\_iterator< \_It > **make\_const\_iterator** (\_It \_\_it) noexcept(is\_nothrow\_convertible\_v< \_It, const←  
\_iterator< \_It > >)
- template<semiregular \_Sent>  
constexpr const\_sentinel< \_Sent > **make\_const\_sentinel** (\_Sent \_\_s) noexcept(is\_nothrow\_convertible\_v< ←  
\_Sent, const\_sentinel< \_Sent > >)
- [error\\_code](#) **make\_error\_code** (errc \_\_e) noexcept
- [error\\_code](#) **make\_error\_code** (future\_errc \_\_errc) noexcept
- [error\\_code](#) **make\_error\_code** (io\_errc \_\_e) noexcept
- [error\\_condition](#) **make\_error\_condition** (errc \_\_e) noexcept
- [error\\_condition](#) **make\_error\_condition** (future\_errc \_\_errc) noexcept
- [error\\_condition](#) **make\_error\_condition** (io\_errc \_\_e) noexcept
- template<typename \_Ex>  
constexpr [exception\\_ptr](#) **make\_exception\_ptr** (\_Ex \_\_ex) noexcept
- template<typename \_RAIter>  
constexpr void **make\_heap** (\_RAIter, \_RAIter)
- template<typename \_RAIter, typename \_Compare>  
constexpr void **make\_heap** (\_RAIter, \_RAIter, \_Compare)
- template<typename \_RandomAccessIterator>  
constexpr void **make\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare>  
constexpr void **make\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_Iterator>  
constexpr [move\\_iterator](#)< \_Iterator > **make\_move\_iterator** (\_Iterator \_\_i)
- template<typename \_Tp, typename \_Alloc, typename... \_Args>  
constexpr \_Tp **make\_obj\_using\_allocator** (const \_Alloc & \_\_a, \_Args &&... \_\_args)
- template<typename \_T1, typename \_T2>  
constexpr [pair](#)< typename \_\_decay\_and\_strip< \_T1 >::\_\_type, typename \_\_decay\_and\_strip< \_T2 >::\_\_type  
> **make\_pair** (\_T1 && \_\_x, \_T2 && \_\_y)
- template<typename \_Iterator>  
constexpr [reverse\\_iterator](#)< \_Iterator > **make\_reverse\_iterator** (\_Iterator \_\_i)
- template<typename... \_Elements>  
constexpr [tuple](#)< typename \_\_decay\_and\_strip< \_Elements >::\_\_type... > **make\_tuple** (\_Elements &&... \_\_←  
args)



- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< __iter_key_t< _InputIterator > >, typename _Allocator = allocator< __iter_key_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >`
- `template<typename _Tp>`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp, typename _Ap>`  
`constexpr basic_vec< _Tp, _Ap > max (const basic_vec< _Tp, _Ap > &__a, const basic_vec< _Tp, _Ap > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CArray>`  
`mdspan (_CArray &) -> mdspan< remove_all_extents_t< _CArray >, extents< size_t, extent_v< _CArray, 0 > > >`
- `template<typename _ElementType, typename... _Integrals>`  
`mdspan (_ElementType *, _Integrals...) -> mdspan< _ElementType, extents< size_t, __detail::__maybe_static_ext< _Integrals >... > >`
- `template<typename _ElementType, typename _MappingType>`  
`mdspan (_ElementType *, const _MappingType &) -> mdspan< _ElementType, typename _MappingType::extents_type, typename _MappingType::layout_type >`
- `template<typename _ElementType, typename _OIndexType, size_t _Nm>`  
`mdspan (_ElementType *, const array< _OIndexType, _Nm > &) -> mdspan< _ElementType, dextents< size_t, _Nm > >`
- `template<typename _ElementType, typename _IndexType, size_t... _ExtentsPack>`  
`mdspan (_ElementType *, const extents< _IndexType, _ExtentsPack... > &) -> mdspan< _ElementType, extents< _IndexType, _ExtentsPack... > >`
- `template<typename _ElementType, typename _OIndexType, size_t _Nm>`  
`mdspan (_ElementType *, span< _OIndexType, _Nm >) -> mdspan< _ElementType, dextents< size_t, _Nm > >`

- `template<typename _Pointer>`  
`mdspan ( _Pointer && ) -> mdspan< remove\_pointer\_t< remove\_reference\_t< _Pointer > >, extents< size_t >`  
`>`
- `template<typename _MappingType, typename _AccessorType>`  
`mdspan (typename _AccessorType::data_handle_type, const _MappingType &, const _AccessorType &) -`  
`> mdspan< typename _AccessorType::element_type, typename _MappingType::extents_type, typename _`  
`MappingType::layout_type, _AccessorType >`
- `template<typename _Tp, typename _Class>`  
`constexpr_Mem_fn< _Tp _Class::* > mem\_fn ( _Tp _Class::* __pm ) noexcept`
- `template<typename _Ret, typename _Tp>`  
`const\_mem\_fun\_t< _Ret, _Tp > mem\_fun ( _Ret( _Tp::* __f )() const)`
- `template<typename _Ret, typename _Tp>`  
`mem\_fun\_t< _Ret, _Tp > mem\_fun ( _Ret( _Tp::* __f )())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const\_mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun ( _Ret( _Tp::* __f )( _Arg ) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem\_fun1\_t< _Ret, _Tp, _Arg > mem\_fun ( _Ret( _Tp::* __f )( _Arg ))`
- `template<typename _Ret, typename _Tp>`  
`const\_mem\_fun\_ref\_t< _Ret, _Tp > mem\_fun\_ref ( _Ret( _Tp::* __f )() const)`
- `template<typename _Ret, typename _Tp>`  
`mem\_fun\_ref\_t< _Ret, _Tp > mem\_fun\_ref ( _Ret( _Tp::* __f )())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const\_mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun\_ref ( _Ret( _Tp::* __f )( _Arg ) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem\_fun1\_ref\_t< _Ret, _Tp, _Arg > mem\_fun\_ref ( _Ret( _Tp::* __f )( _Arg ))`
- `void * memchr (void * __s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr_OIter merge ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr_OIter merge ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator merge ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _`  
`InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator merge ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _`  
`InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr const _Tp & min (const _Tp & __a, const _Tp & __b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & min (const _Tp & __a, const _Tp & __b, _Compare __comp)`
- `template<typename _Tp, typename _Ap>`  
`constexpr basic_vec< _Tp, _Ap > min (const basic_vec< _Tp, _Ap > & __a, const basic_vec< _Tp, _Ap >`  
`& __b) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp min (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp min (initializer\_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr_Filter min\_element ( _Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr_Filter min\_element ( _Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator min\_element ( _ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp, typename _Ap>`  
`constexpr pair< basic_vec< _Tp, _Ap >, basic_vec< _Tp, _Ap > > minmax (const basic_vec< _Tp, _Ap > &__a, const basic_vec< _Tp, _Ap > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > minmax (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > minmax (initializer\_list< _Tp >, _Compare)`
- `template<typename _Filter>`  
`constexpr pair< _Filter, _Filter > minmax\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr pair< _Filter, _Filter > minmax\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `float modf (float __x, float * __iptr)`
- `long double modf (long double __x, long double * __iptr)`
- `template<typename _II, typename _OI>`  
`constexpr _OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp>`  
`constexpr std::remove\_reference< _Tp >::type && move (_Tp &&__t) noexcept`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > move\_if\_noexcept (_Tp &__x) noexcept`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`

- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multimap` (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multimap`< `__iter_key_t<_InputIterator>`, `__iter_val_t<_InputIterator>`, `_Compare`, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`multimap` (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `_Allocator`) -> `multimap`< `_Key`, `_Tp`, `less`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multimap` (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multimap`< `_Key`, `_Tp`, `_Compare`, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`multiset` (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `multiset`< `typename iterator_traits<_InputIterator>::value_type`, `less`< `typename iterator_traits<_InputIterator>::value_type` >, `_Allocator` >
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multiset` (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset`< `typename iterator_traits<_InputIterator>::value_type`, `_Compare`, `_Allocator` >
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`multiset` (`initializer_list`< `_Key` >, `_Allocator`) -> `multiset`< `_Key`, `less`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`multiset` (`initializer_list`< `_Key` >, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset`< `_Key`, `_Compare`, `_Allocator` >
- `template<typename _Tp>`  
`constexpr` `__gnu_cxx::enable_if<__is_integer<_Tp>::value, double>::type` `nearbyint` (`_Tp __x`)
- `constexpr` `float` `nearbyint` (`float __x`)
- `constexpr` `long double` `nearbyint` (`long double __x`)
- `template<typename _InputIterator>`  
`constexpr` `_InputIterator` `next` (`_InputIterator __x`, `typename iterator_traits<_InputIterator>::difference_type __n=1`)
- `template<typename _BidirectionalIterator>`  
`constexpr` `bool` `next_permutation` (`_BidirectionalIterator __first`, `_BidirectionalIterator __last`)
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr` `bool` `next_permutation` (`_BidirectionalIterator __first`, `_BidirectionalIterator __last`, `_Compare __comp`)
- `template<typename _Biter>`  
`constexpr` `bool` `next_permutation` (`_Biter`, `_Biter`)
- `template<typename _Biter, typename _Compare>`  
`constexpr` `bool` `next_permutation` (`_Biter`, `_Biter`, `_Compare`)
- `template<typename _Tp, typename _Up>`  
`constexpr` `__gnu_cxx::promote_2<_Tp, _Up>::type` `nextafter` (`_Tp __x`, `_Up __y`)
- `constexpr` `float` `nextafter` (`float __x`, `float __y`)
- `constexpr` `long double` `nextafter` (`long double __x`, `long double __y`)
- `template<typename _Tp>`  
`constexpr` `__gnu_cxx::enable_if<__is_integer<_Tp>::value, double>::type` `nexttoward` (`_Tp __x`, `long double __y`)
- `constexpr` `float` `nexttoward` (`float __x`, `long double __y`)
- `constexpr` `long double` `nexttoward` (`long double __x`, `long double __y`)
- `ios_base` & `noboolalpha` (`ios_base` & `__base`)

- `template<typename _Iter, typename _Predicate>`  
`constexpr bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type norm (_Tp __x)`
- `template<typename _Tp>`  
`_Tp constexpr norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr _Tp norm (const complex< _Tp > &__z)`
- `ios_base & noshowbase (ios_base &__base)`
- `ios_base & noshowpoint (ios_base &__base)`
- `ios_base & noshowpos (ios_base &__base)`
- `ios_base & noskipws (ios_base &__base)`
- `template<typename _Predicate>`  
`constexpr unary_negate< _Predicate > not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`  
`constexpr binary_negate< _Predicate > not2 (const _Predicate &__pred)`
- `void notify_all_at_thread_exit (condition_variable &, unique_lock< mutex >)`
- `ios_base & nouitbuf (ios_base &__base)`
- `ios_base & nouppercase (ios_base &__base)`
- `template<typename _RAlter>`  
`constexpr void nth_element (_RAlter, _RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare>`  
`constexpr void nth_element (_RAlter, _RAlter, _RAlter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base &__base)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _StateT>`  
`bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Tp, typename _Seq>`  
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > operator!= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__modulus, typename _Dom1::value_type >::result_type > operator% (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr _ios_Fmtflags operator& (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_ostate operator& (_ios_ostate __a, _ios_ostate __b) noexcept`
- `constexpr _ios_Openmode operator& (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr chars_format operator& (chars_format __lhs, chars_format __rhs) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__bitwise_and, typename _Dom1::value_type >::result_type > operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const type-`  
`name _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const`  
`valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `constexpr launch operator& (launch __x, launch __y) noexcept`
- `constexpr memory\_order operator& (memory\_order __m, __memory_order_modifier __mod) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _`  
`Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const _`  
`Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`logical_and, typename _Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const type-`  
`name _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray<`  
`typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `constexpr const _los_Fmtflags & operator&= (_los_Fmtflags & __a, _los_Fmtflags __b) noexcept`
- `constexpr const _los_losestate & operator&= (_los_losestate & __a, _los_losestate __b) noexcept`
- `constexpr const _los_Openmode & operator&= (_los_Openmode & __a, _los_Openmode __b) noexcept`
- `constexpr chars\_format & operator&= (chars\_format & __lhs, chars\_format __rhs) noexcept`
- `constexpr launch & operator&= (launch & __x, launch __y) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`multiplies, typename _Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _`  
`Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const typename _`  
`Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`



- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _`  
`_Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`_CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, basic_string< _CharT, _`  
`_Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT,`  
`_Traits, _Alloc > &__rhs)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _`  
`_fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _`  
`_fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus,`  
`typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_`  
`type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &`  
`__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc >`  
`&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &`  
`__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &`  
`__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`



- `template<typename _Tp>`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _<←`  
`_fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::<←`  
`value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _<←`  
`_fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() + __n } -> same_as<_Iterator>; }`  
`constexpr move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n,`  
`const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator>`  
`constexpr reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type <←`  
`__n, const reverse_iterator< _Iterator > &__x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _<←`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _<←`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__minus,`  
`typename _Dom1::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type`  
`> &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator- (const complex< _Tp > &__x)`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->`  
`decltype(__x.base() - __y.base())`
- `template<typename _IteratorL, typename _IteratorR>`  
`constexpr auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`  
`-> decltype(__y.base() - __x.base())`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom::<←`  
`value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _<←`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename <←`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↵`  
`_fun< struct std::_less, typename _Dom::value_type >::result_type > operator< (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator< (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, const error\_code`  
`&__e)`
- `template<typename _Ostream, typename _Tp>`  
`_rvalue_stream_insertion_t< _Ostream, _Tp > operator<< (_Ostream &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_money<`  
`_MoneyT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_time< _↵`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setfill< _CharT`  
`> __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, basic\_string\_view<`  
`_CharT, _Traits > __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, const`  
`\_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, const`  
`basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic\_ostream< _CharT, _Traits > & operator<< (basic\_ostream< _CharT, _Traits > &__os, const complex<`  
`_Tp > &__x)`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & operator<< (basic\_ostream< char, _Traits > &, char16\_t)=delete`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & operator<< (basic\_ostream< char, _Traits > &, char32\_t)=delete`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & operator<< (basic\_ostream< char, _Traits > &, const char16\_t *)=delete`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & operator<< (basic\_ostream< char, _Traits > &, const char32\_t *)=delete`

- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`_shift_left, typename _Dom1::value_type >::result_type > operator<< (const _Expr< _Dom1, typename _`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const typename _`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template ostream & operator<< (ostream &, _Resetiosflags)`
- `template ostream & operator<< (ostream &, _Setbase)`
- `template ostream & operator<< (ostream &, _Setfill< char >)`
- `template ostream & operator<< (ostream &, _Setiosflags)`
- `template ostream & operator<< (ostream &, _Setprecision)`
- `template ostream & operator<< (ostream &, _Setw)`
- `template ostream & operator<< (ostream &, const complex< double > &)`
- `template ostream & operator<< (ostream &, const complex< float > &)`
- `template ostream & operator<< (ostream &, const complex< long double > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`binomial_distribution< _IntType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`chi_squared_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`lognormal_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, ↵`  
`_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template wostream & operator<< (wostream &, _Resetiosflags)`
- `template wostream & operator<< (wostream &, _Setbase)`
- `template wostream & operator<< (wostream &, _Setfill< wchar_t >)`
- `template wostream & operator<< (wostream &, _Setiosflags)`
- `template wostream & operator<< (wostream &, _Setprecision)`
- `template wostream & operator<< (wostream &, _Setw)`
- `template wostream & operator<< (wostream &, const complex< double > &)`
- `template wostream & operator<< (wostream &, const complex< float > &)`
- `template wostream & operator<< (wostream &, const complex< long double > &)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr<`  
`_Tp, _Lp > &__p)`
- `template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr<`  
`_Tp, _Dp > &__p)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _↵`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _↵`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__↵`  
`_less_equal, typename _Dom1::value_type >::result_type > operator<= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↵`  
`_Compare, _Alloc > &__y)`



- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() } >= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const typename <`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept`
- `strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr auto operator<=> (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view<`  
`_CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>`  
`strong_ordering operator<=> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b)`  
`noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`strong_ordering operator<=> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const array< _Tp, _Nm > &__a, const array< _Tp,`  
`_Nm > &__b)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs) noex-`  
`cept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT,`  
`_Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`  
`> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list<`  
`_Tp, _Alloc > &__y)`

- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t< _Tp > operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare_three_way_result_t< _Iterator > operator<=> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare_three_way_result_t< _IteratorL, _IteratorR > operator<=> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > operator<=> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare_three_way_result_t< _Iterator, _Iterator > operator<=> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare_three_way_result_t< _IteratorL, _IteratorR > operator<=> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > operator<=> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename... _Tps, typename... _Ups>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > operator<=> (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > operator<=> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > operator<=> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `constexpr strong_ordering operator<=> (monostate, monostate) noexcept`
- `template<typename _T1, typename _T2>`  
`constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept`
- `template<typename _Bi_iter, typename _Alloc>`  
`bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`



- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__equal_to, typename _Dom1::value_type >::result_type > operator== (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool operator== (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool operator== (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const basic\_string< _CharT,`  
`_Traits, _Alloc > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _StateT>`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _CharT, typename _Traits>`  
`bool operator== (const istreambuf\_iterator< _CharT, _Traits > &__a, const istreambuf\_iterator< _CharT, _Traits`  
`> &__b)`
- `template<typename _Tp, typename _Alloc>`  
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc`  
`> &__y)`
- `template<typename _Iterator>`  
`constexpr bool operator== (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _RealType>`  
`bool operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename... _Tps, typename... _Ups>`  
`constexpr bool operator== (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `constexpr bool operator== (monostate, monostate) noexcept`
- `template<typename _OutA1, typename _OutA2, typename... _InA>`  
`bool operator== (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp, typename _Dp>`  
`bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Istream, typename _Tp>`  
`_rvalue_stream_extraction_t< _Istream, _Tp > operator>> (_Istream &&__is, _Tp &&__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string<`  
`_CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)`

- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<>` `basic_istream< char > & operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_right, typename _Dom1::value_type >::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template` `istream & operator>> (istream &, _Resetiosflags)`
- `template` `istream & operator>> (istream &, _Setbase)`
- `template` `istream & operator>> (istream &, _Setfill< char >)`
- `template` `istream & operator>> (istream &, _Setiosflags)`
- `template` `istream & operator>> (istream &, _Setprecision)`
- `template` `istream & operator>> (istream &, _Setw)`
- `template` `istream & operator>> (istream &, complex< double > &)`
- `template` `istream & operator>> (istream &, complex< float > &)`
- `template` `istream & operator>> (istream &, complex< long double > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)`

- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`lognormal_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,`  
`_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`&__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::bernoulli_distribution &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::extreme_value_distribution< _RealType > &__x)`



- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template wistream & operator>> (wistream &, _Resetiosflags)`
- `template wistream & operator>> (wistream &, _Setbase)`
- `template wistream & operator>> (wistream &, _Setfill< wchar_t >)`
- `template wistream & operator>> (wistream &, _Setiosflags)`
- `template wistream & operator>> (wistream &, _Setprecision)`
- `template wistream & operator>> (wistream &, _Setw)`
- `template wistream & operator>> (wistream &, complex< double > &)`
- `template wistream & operator>> (wistream &, complex< float > &)`
- `template wistream & operator>> (wistream &, complex< long double > &)`
- `constexpr _los_Fmtflags operator^ (_los_Fmtflags __a, _los_Fmtflags __b) noexcept`
- `constexpr _los_losestate operator^ (_los_losestate __a, _los_losestate __b) noexcept`
- `constexpr _los_Openmode operator^ (_los_Openmode __a, _los_Openmode __b) noexcept`
- `constexpr chars_format operator^ (chars_format __lhs, chars_format __rhs) noexcept`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__bitwise_xor, typename _Dom1::value_type >::result_type > operator^ (const _Expr< _Dom1, typename _`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr launch operator^ (launch __x, launch __y) noexcept`
- `constexpr const _los_Fmtflags & operator^= (_los_Fmtflags &__a, _los_Fmtflags __b) noexcept`
- `constexpr const _los_losestate & operator^= (_los_losestate &__a, _los_losestate __b) noexcept`
- `constexpr const _los_Openmode & operator^= (_los_Openmode &__a, _los_Openmode __b) noexcept`
- `constexpr chars_format & operator^= (chars_format &__lhs, chars_format __rhs) noexcept`
- `constexpr launch & operator^= (launch &__x, launch __y) noexcept`
- `constexpr _los_Fmtflags operator| (_los_Fmtflags __a, _los_Fmtflags __b) noexcept`

- constexpr **\_los\_losestate operator|** (**\_los\_losestate** \_\_a, **\_los\_losestate** \_\_b) noexcept
- constexpr **\_los\_Openmode operator|** (**\_los\_Openmode** \_\_a, **\_los\_Openmode** \_\_b) noexcept
- constexpr **chars\_format operator|** (**chars\_format** \_\_lhs, **chars\_format** \_\_rhs) noexcept
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator|** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e, const **valarray**< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator|** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom1, class \_Dom2>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< struct std::\_\_  
\_\_bitwise\_or, typename \_Dom1::value\_type >::result\_type > **operator|** (const \_Expr< \_Dom1, typename \_  
Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator|** (const typename \_  
Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_bitwise\_or, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **operator|** (const **valarray**< type-  
name \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- constexpr **launch operator|** (**launch** \_\_x, **launch** \_\_y) noexcept
- constexpr **memory\_order operator|** (**memory\_order** \_\_m, \_\_memory\_order\_modifier \_\_mod) noexcept
- constexpr const **\_los\_Fmtflags & operator|=** (**\_los\_Fmtflags** &\_\_a, **\_los\_Fmtflags** \_\_b) noexcept
- constexpr const **\_los\_losestate & operator|=** (**\_los\_losestate** &\_\_a, **\_los\_losestate** \_\_b) noexcept
- constexpr const **\_los\_Openmode & operator|=** (**\_los\_Openmode** &\_\_a, **\_los\_Openmode** \_\_b) noexcept
- constexpr **chars\_format & operator|=** (**chars\_format** &\_\_lhs, **chars\_format** \_\_rhs) noexcept
- constexpr **launch & operator|=** (**launch** &\_\_x, **launch** \_\_y) noexcept
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_logical\_or, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_logical\_or, typename \_Dom::value\_type >::result\_type > **operator||** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e, const **valarray**< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_logical\_or, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename  
\_\_fun< struct std::\_\_logical\_or, typename \_Dom::value\_type >::result\_type > **operator||** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom1, class \_Dom2>  
\_Expr< \_BinClos< struct std::\_\_logical\_or, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< struct std::\_\_  
\_\_logical\_or, typename \_Dom1::value\_type >::result\_type > **operator||** (const \_Expr< \_Dom1, typename \_  
Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_logical\_or, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_logical\_or, typename \_Dom::value\_type >::result\_type > **operator||** (const typename \_  
Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom>  
\_Expr< \_BinClos< struct std::\_\_logical\_or, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename  
\_\_fun< struct std::\_\_logical\_or, typename \_Dom::value\_type >::result\_type > **operator||** (const **valarray**< type-  
name \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- constexpr **\_los\_Fmtflags operator~** (**\_los\_Fmtflags** \_\_a) noexcept
- constexpr **\_los\_losestate operator~** (**\_los\_losestate** \_\_a) noexcept
- constexpr **\_los\_Openmode operator~** (**\_los\_Openmode** \_\_a) noexcept



- constexpr **chars\_format operator~** (**chars\_format** \_\_fmt) noexcept
- constexpr **launch operator~** (**launch** \_\_x) noexcept
- template<typename \_Pointer = void, typename \_Smart, typename... \_Args>  
auto **out\_ptr** (\_Smart &\_\_s, \_Args &&... \_\_args)
- template<typename \_Fun, typename \_Signature = \_\_function\_guide\_t<\_Fun, decltype(&\_Fun::operator())>>  
**packaged\_task** (\_Fun) -> packaged\_task<\_Signature >
- template<typename \_Res, typename... \_ArgTypes>  
**packaged\_task** (\_Res(\*)(\_ArgTypes...)) -> packaged\_task<\_Res(\_ArgTypes...)>
- template<typename \_RAlter>  
constexpr void **partial\_sort** (\_RAlter, \_RAlter, \_RAlter)
- template<typename \_RAlter, typename \_Compare>  
constexpr void **partial\_sort** (\_RAlter, \_RAlter, \_RAlter, \_Compare)
- template<typename \_RandomAccessIterator>  
constexpr void **partial\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_middle, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare>  
constexpr void **partial\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_middle, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_IIter, typename \_RAlter>  
constexpr \_RAlter **partial\_sort\_copy** (\_IIter, \_IIter, \_RAlter, \_RAlter)
- template<typename \_IIter, typename \_RAlter, typename \_Compare>  
constexpr \_RAlter **partial\_sort\_copy** (\_IIter, \_IIter, \_RAlter, \_RAlter, \_Compare)
- template<typename \_InputIterator, typename \_RandomAccessIterator>  
constexpr \_RandomAccessIterator **partial\_sort\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_\_result\_first, \_RandomAccessIterator \_\_result\_last)
- template<typename \_InputIterator, typename \_RandomAccessIterator, typename \_Compare>  
constexpr \_RandomAccessIterator **partial\_sort\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_RandomAccessIterator \_\_result\_first, \_RandomAccessIterator \_\_result\_last, \_Compare \_\_comp)
- template<typename \_InputIterator, typename \_OutputIterator>  
constexpr \_OutputIterator **partial\_sum** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_BinaryOperation>  
constexpr \_OutputIterator **partial\_sum** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename \_BIter, typename \_Predicate>  
constexpr \_BIter **partition** (\_BIter, \_BIter, \_Predicate)
- template<typename \_ForwardIterator, typename \_Predicate>  
constexpr \_ForwardIterator **partition** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_IIter, typename \_OIIter1, typename \_OIIter2, typename \_Predicate>  
constexpr pair<\_OIIter1, \_OIIter2> **partition\_copy** (\_IIter, \_IIter, \_OIIter1, \_OIIter2, \_Predicate)
- template<typename \_InputIterator, typename \_OutputIterator1, typename \_OutputIterator2, typename \_Predicate>  
constexpr pair<\_OutputIterator1, \_OutputIterator2> **partition\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator1 \_\_out\_true, \_OutputIterator2 \_\_out\_false, \_Predicate \_\_pred)
- template<typename \_Filter, typename \_Predicate>  
constexpr \_Filter **partition\_point** (\_Filter, \_Filter, \_Predicate)
- template<typename \_ForwardIterator, typename \_Predicate>  
constexpr \_ForwardIterator **partition\_point** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_Tp>  
**complex**<\_Tp> **polar** (const \_Tp &, const \_Tp &= \_Tp(0))
- template<typename \_RAlter>  
constexpr void **pop\_heap** (\_RAlter, \_RAlter)
- template<typename \_RAlter, typename \_Compare>  
constexpr void **pop\_heap** (\_RAlter, \_RAlter, \_Compare)

- `template<typename _RandomAccessIterator>`  
`constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > pow (const complex< _Tp > &, int)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `constexpr float pow (float __x, float __y)`

- constexpr long double **pow** (long double \_\_x, long double \_\_y)
- template<typename \_BidirectionalIterator>  
constexpr \_BidirectionalIterator **prev** (\_BidirectionalIterator \_\_x, typename iterator\_traits<\_BidirectionalIterator>::difference\_type \_\_n=1)
- template<typename \_BidirectionalIterator>  
constexpr bool **prev\_permutation** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_last)
- template<typename \_BidirectionalIterator, typename \_Compare>  
constexpr bool **prev\_permutation** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_last, \_Compare \_\_comp)
- template<typename \_BIter>  
constexpr bool **prev\_permutation** (\_BIter, \_BIter)
- template<typename \_BIter, typename \_Compare>  
constexpr bool **prev\_permutation** (\_BIter, \_BIter, \_Compare)
- template<typename \_Compare, typename \_Container, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireNotAllocator<\_Container>>>  
**priority\_queue** (\_Compare, \_Container) -> priority\_queue< typename \_Container::value\_type, \_Container, \_Compare >
- template<typename \_Compare, typename \_Container, typename \_Allocator, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireNotAllocator<\_Container>>>  
**priority\_queue** (\_Compare, \_Container, \_Allocator) -> priority\_queue< typename \_Container::value\_type, \_Container, \_Compare >
- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Compare = less<\_ValT>, typename \_Container = vector<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireNotAllocator<\_Container>>>  
**priority\_queue** (\_InputIterator, \_InputIterator, \_Compare=\_Compare(), \_Container=\_Container()) -> priority\_queue< \_ValT, \_Container, \_Compare >
- template<typename \_Tp>  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote<\_Tp>::\_\_type > **proj** (\_Tp \_\_x)
- template<typename \_Tp>  
**std::complex**< \_Tp > **proj** (const **std::complex**< \_Tp > &)
- template<typename \_Arg, typename \_Result>  
**pointer\_to\_unary\_function**< \_Arg, \_Result > **ptr\_fun** (\_Result(\*\_\_x)(\_Arg))
- template<typename \_Arg1, typename \_Arg2, typename \_Result>  
**pointer\_to\_binary\_function**< \_Arg1, \_Arg2, \_Result > **ptr\_fun** (\_Result(\*\_\_x)(\_Arg1, \_Arg2))
- template<typename \_RAIter>  
constexpr void **push\_heap** (\_RAIter, \_RAIter)
- template<typename \_RAIter, typename \_Compare>  
constexpr void **push\_heap** (\_RAIter, \_RAIter, \_Compare)
- template<typename \_RandomAccessIterator>  
constexpr void **push\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare>  
constexpr void **push\_heap** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_MoneyT>  
\_Put\_money< \_MoneyT > **put\_money** (const \_MoneyT &\_\_mon, bool \_\_intl=false)
- template<typename \_CharT>  
\_Put\_time< \_CharT > **put\_time** (const std::tm \* \_\_tmb, const \_CharT \* \_\_fmt)
- template<typename \_Container, typename = \_RequireNotAllocator<\_Container>>>  
**queue** (\_Container) -> queue< typename \_Container::value\_type, \_Container >
- template<typename \_Container, typename \_Allocator, typename = \_RequireNotAllocator<\_Container>>>  
**queue** (\_Container, \_Allocator) -> queue< typename \_Container::value\_type, \_Container >
- template<typename \_RAIter>  
void **random\_shuffle** (\_RAIter, \_RAIter)
- template<typename \_RAIter, typename \_Generator>  
void **random\_shuffle** (\_RAIter, \_RAIter, \_Generator &&)

- `template<typename _RandomAccessIterator>`  
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _Container>`  
`constexpr auto rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > rbegin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())`
- `template<typename _Tp>`  
`constexpr reverse_iterator< const _Tp * > rbegin (initializer_list< _Tp > __il) noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp>`  
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _InputIterator>`  
`constexpr iterator_traits< _InputIterator >::value_type reduce (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Tp>`  
`reference_wrapper (_Tp &) -> reference_wrapper< _Tp >`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > reinterpret_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type remainder (_Tp __x, _Up __y)`
- `constexpr float remainder (float __x, float __y)`
- `constexpr long double remainder (long double __x, long double __y)`
- `template<typename _Filter, typename _Tp = typename iterator_traits< _Filter >::value_type>`  
`constexpr _Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp = typename iterator_traits< _Iter >::value_type>`  
`constexpr _OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Up>`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type remquo (_Tp __x, _Up __y, int *__pquo)`

- float **remquo** (float \_\_x, float \_\_y, int \*\_\_pquo)
- long double **remquo** (long double \_\_x, long double \_\_y, int \*\_\_pquo)
- template<typename \_Container>  
constexpr auto **rend** (\_Container &\_\_cont) noexcept(noexcept(\_\_cont.rend())) -> decltype(\_\_cont.rend())
- template<typename \_Tp, size\_t \_Nm>  
constexpr **reverse\_iterator**< \_Tp \* > **rend** (\_Tp(&\_\_arr)[\_Nm]) noexcept
- template<typename \_Container>  
constexpr auto **rend** (const \_Container &\_\_cont) noexcept(noexcept(\_\_cont.rend())) -> decltype(\_\_cont.rend())
- template<typename \_Tp>  
constexpr **reverse\_iterator**< const \_Tp \* > **rend** (initializer\_list< \_Tp > \_\_il) noexcept
- template<typename \_Filter, typename \_Tp = typename iterator\_traits<\_Filter>::value\_type>  
constexpr void **replace** (\_Filter, \_Filter, const \_Tp &, const \_Tp &)
- template<typename \_ForwardIterator, typename \_Tp>  
constexpr void **replace** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value)
- template<typename \_Iter, typename \_OIter, typename \_Tp>  
constexpr \_OIter **replace\_copy** (\_Iter, \_Iter, \_OIter, const \_Tp &, const \_Tp &)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_Tp>  
constexpr \_OutputIterator **replace\_copy** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_Predicate, typename \_Tp>  
constexpr \_OutputIterator **replace\_copy\_if** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, \_Predicate \_\_pred, const \_Tp &\_\_new\_value)
- template<typename \_Iter, typename \_OIter, typename \_Predicate, typename \_Tp = typename iterator\_traits<\_OIter>::value\_type>  
constexpr \_OIter **replace\_copy\_if** (\_Iter, \_Iter, \_OIter, \_Predicate, const \_Tp &)
- template<typename \_Filter, typename \_Predicate, typename \_Tp = typename iterator\_traits<\_Filter>::value\_type>  
constexpr void **replace\_if** (\_Filter, \_Filter, \_Predicate, const \_Tp &)
- template<typename \_ForwardIterator, typename \_Predicate, typename \_Tp>  
constexpr void **replace\_if** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Predicate \_\_pred, const \_Tp &\_\_new\_value)
- \_Resetiosflags **resetiosflags** (ios\_base::fmtflags \_\_mask)
- void **rethrow\_exception** (exception\_ptr)
- template<typename \_Ex>  
void **rethrow\_if\_nested** (const \_Ex &\_\_ex)
- template<typename \_Tp>  
void **return\_temporary\_buffer** (\_Tp \* \_\_p)
- template<typename \_BidirectionalIterator>  
constexpr void **reverse** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_last)
- template<typename \_BIter>  
constexpr void **reverse** (\_BIter, \_BIter)
- template<typename \_BidirectionalIterator, typename \_OutputIterator>  
constexpr \_OutputIterator **reverse\_copy** (\_BidirectionalIterator \_\_first, \_BidirectionalIterator \_\_last, \_OutputIterator \_\_result)
- template<typename \_BIter, typename \_OIter>  
constexpr \_OIter **reverse\_copy** (\_BIter, \_BIter, \_OIter)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **riemann\_zeta** (\_Tp \_\_s)
- float **riemann\_zetaf** (float \_\_s)
- long double **riemann\_zetal** (long double \_\_s)
- ios\_base & **right** (ios\_base &\_\_base)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **rint** (\_Tp \_\_x)

- constexpr float **rint** (float \_\_x)
- constexpr long double **rint** (long double \_\_x)
- template<typename \_Filter>  
constexpr \_Filter **rotate** (\_Filter, \_Filter, \_Filter)
- template<typename \_ForwardIterator>  
constexpr \_ForwardIterator **rotate** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_middle, \_ForwardIterator \_\_last)
- template<typename \_Filter, typename \_OIter>  
constexpr \_OIter **rotate\_copy** (\_Filter, \_Filter, \_Filter, \_OIter)
- template<typename \_ForwardIterator, typename \_OutputIterator>  
constexpr \_OutputIterator **rotate\_copy** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_middle, \_ForwardIterator \_\_last, \_OutputIterator \_\_result)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **round** (\_Tp \_\_x)
- constexpr float **round** (float \_\_x)
- constexpr long double **round** (long double \_\_x)
- template<typename \_PopulationIterator, typename \_SampleIterator, typename \_Distance, typename \_UniformRandomBitGenerator>  
\_SampleIterator **sample** (\_PopulationIterator \_\_first, \_PopulationIterator \_\_last, \_SampleIterator \_\_out, \_Distance \_\_n, \_UniformRandomBitGenerator &&\_g)
- template<typename \_Tp>  
constexpr \_Tp **saturating\_add** (\_Tp \_\_x, \_Tp \_\_y) noexcept
- template<typename \_Res, typename \_Tp>  
constexpr \_Res **saturating\_cast** (\_Tp \_\_x) noexcept
- template<typename \_Tp>  
constexpr \_Tp **saturating\_div** (\_Tp \_\_x, \_Tp \_\_y) noexcept
- template<typename \_Tp>  
constexpr \_Tp **saturating\_mul** (\_Tp \_\_x, \_Tp \_\_y) noexcept
- template<typename \_Tp>  
constexpr \_Tp **saturating\_sub** (\_Tp \_\_x, \_Tp \_\_y) noexcept
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **scalbn** (\_Tp \_\_x, long \_\_ex)
- constexpr float **scalbn** (float \_\_x, long \_\_ex)
- constexpr long double **scalbn** (long double \_\_x, long \_\_ex)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **scalbn** (\_Tp \_\_x, int \_\_ex)
- constexpr float **scalbn** (float \_\_x, int \_\_ex)
- constexpr long double **scalbn** (long double \_\_x, int \_\_ex)
- **ios\_base** & **scientific** (**ios\_base** & \_\_base)
- template<typename \_Filter1, typename \_Filter2>  
constexpr \_Filter1 **search** (\_Filter1, \_Filter1, \_Filter2, \_Filter2)
- template<typename \_Filter1, typename \_Filter2, typename \_BinaryPredicate>  
constexpr \_Filter1 **search** (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_BinaryPredicate)
- template<typename \_ForwardIterator, typename \_Searcher>  
constexpr \_ForwardIterator **search** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Searcher & \_\_searcher)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2>  
constexpr \_ForwardIterator1 **search** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_ForwardIterator2 \_\_last2)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2, typename \_BinaryPredicate>  
constexpr \_ForwardIterator1 **search** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2, \_ForwardIterator2 \_\_last2, \_BinaryPredicate \_\_predicate)
- template<typename \_Filter, typename \_Size, typename \_Tp = typename iterator\_traits<\_Filter>::value\_type>  
constexpr \_Filter **search\_n** (\_Filter, \_Filter, \_Size, const \_Tp &)

- `template<typename _Filter, typename _Size, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _BinaryPredicate> constexpr _Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp> constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate> constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>> set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>> set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>> set (initializer_list<_Key >, _Allocator) -> set< _Key, less<_Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>> set (initializer_list<_Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Iter1, typename _Iter2, typename _OIter> constexpr _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> constexpr _OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter> constexpr _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> constexpr _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `new_handler set_new_handler (new_handler) throw ()`
- `template<typename _Iter1, typename _Iter2, typename _OIter> constexpr _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare> constexpr _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`



- `terminate_handler set_terminate (terminate_handler)` noexcept
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT>`  
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `template<typename _Tp, typename _Del>`  
`shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >`
- `template<typename _Tp>`  
`shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator shift_left (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits<  
_ForwardIterator >::difference_type __n)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator shift_right (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits<  
_ForwardIterator >::difference_type __n)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RAIter, typename _UGenerator>`  
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`  
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator  
&& __g)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type signbit (_Tp __x)`
- `constexpr bool signbit (double __x)`
- `constexpr bool signbit (float __x)`
- `constexpr bool signbit (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::Sin, _Expr, _Dom >, typename _Dom::value_type > sin (const _Expr< _Dom,  
typename _Dom::value_type > & __e)`
- `template<typename _Tp>`  
`complex< _Tp > sin (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp > & __v)`
- `constexpr float sin (float __x)`
- `constexpr long double sin (long double __x)`



- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sinh (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > sinh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp > &__v)`
- `constexpr float sinh (float __x)`
- `constexpr long double sinh (long double __x)`
- `template<typename _Container>`  
`constexpr auto size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t size (const _Tp(&)[_Nm]) noexcept`
- `ios_base & skipws (ios_base &__base)`
- `template<typename _RAlter>`  
`constexpr void sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare>`  
`constexpr void sort (_RAlter, _RAlter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAlter>`  
`constexpr void sort_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare>`  
`constexpr void sort_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sqrt (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`complex< _Tp > sqrt (const complex< _Tp > &)`

- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > &__v)`
- `constexpr float sqrt (float __x)`
- `constexpr long double sqrt (long double __x)`
- `template<typename _Container>`  
`constexpr auto ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common\_type\_t< ptrdiff_t, make\_signed\_t< decltype(__cont.size())> >`
- `template<typename _Tp, ptrdiff_t _Num>`  
`constexpr ptrdiff_t ssize (const _Tp(&)[_Num]) noexcept`
- `template<typename _Blter, typename _Predicate>`  
`constexpr _Blter stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RAIter>`  
`constexpr void stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator>`  
`constexpr void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>`  
`stack (_Container) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>`  
`stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Tp>`  
`const _Tp * start_lifetime_as (const void *__p) noexcept`
- `template<typename _Tp>`  
`const volatile _Tp * start_lifetime_as (const volatile void *__p) noexcept`
- `template<typename _Tp>`  
`_Tp * start_lifetime_as (void *__p) noexcept`
- `template<typename _Tp>`  
`volatile _Tp * start_lifetime_as (volatile void *__p) noexcept`
- `template<typename _Tp>`  
`const _Tp * start_lifetime_as_array (const void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`const volatile _Tp * start_lifetime_as_array (const volatile void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`_Tp * start_lifetime_as_array (void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`volatile _Tp * start_lifetime_as_array (volatile void *__p, size_t __n) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `double stod (const string &__str, size_t *__idx=0)`
- `double stod (const wstring &__str, size_t *__idx=0)`
- `float stof (const string &__str, size_t *__idx=0)`
- `float stof (const wstring &__str, size_t *__idx=0)`
- `int stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `int stoi (const wstring &__str, size_t *__idx=0, int __base=10)`
- `long stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long stol (const wstring &__str, size_t *__idx=0, int __base=10)`
- `long double stold (const string &__str, size_t *__idx=0)`

- long double **stold** (const [wstring](#) &\_\_str, size\_t \*\_\_idx=0)
- long long **stoll** (const [string](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- long long **stoll** (const [wstring](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long **stoul** (const [string](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long **stoul** (const [wstring](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long long **stoull** (const [string](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- unsigned long long **stoull** (const [wstring](#) &\_\_str, size\_t \*\_\_idx=0, int \_\_base=10)
- char \* **strchr** (char \*\_\_s, int \_\_n)
- char \* **strpbrk** (char \*\_\_s1, const char \*\_\_s2)
- char \* **strchr** (char \*\_\_s, int \_\_n)
- char \* **strstr** (char \*\_\_s1, const char \*\_\_s2)
- template<typename \_IndexType, size\_t... \_Extents, typename... \_RawSlices>  
constexpr auto **subextents** (const extents< \_IndexType, \_Extents... > &\_\_exts, \_RawSlices... \_\_raw\_slices)
- template<typename \_ElementType, typename \_Extents, typename \_Layout, typename \_Accessor, typename... \_RawSlices>  
constexpr auto **submdspan** (const mdspan< \_ElementType, \_Extents, \_Layout, \_Accessor > &\_\_md, \_RawSlices... \_\_raw\_slices)
- template<typename \_Ch\_type, typename \_Rx\_traits>  
void **swap** ([basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_lhs, [basic\\_regex](#)< \_Ch\_type, \_Rx\_traits > &\_\_rhs) noexcept
- template<typename \_Bi\_iter, typename \_Alloc>  
void **swap** ([match\\_results](#)< \_Bi\_iter, \_Alloc > &\_\_lhs, [match\\_results](#)< \_Bi\_iter, \_Alloc > &\_\_rhs) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
void **swap** (\_\_shared\_ptr< \_Tp, \_Lp > &\_\_a, \_\_shared\_ptr< \_Tp, \_Lp > &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
void **swap** (\_\_weak\_ptr< \_Tp, \_Lp > &\_\_a, \_\_weak\_ptr< \_Tp, \_Lp > &\_\_b) noexcept
- template<typename \_Key, typename \_Val, typename \_KeyOfValue, typename \_Compare, typename \_Alloc>  
void **swap** (\_Rb\_tree< \_Key, \_Val, \_KeyOfValue, \_Compare, \_Alloc > &\_\_x, \_Rb\_tree< \_Key, \_Val, \_KeyOfValue, \_Compare, \_Alloc > &\_\_y)
- template<typename \_Tp>  
constexpr \_Require< \_\_not\_< \_\_is\_tuple\_like< \_Tp > >, [is\\_move\\_constructible](#)< \_Tp >, [is\\_move\\_assignable](#)< \_Tp > > **swap** (\_Tp &, \_Tp &) noexcept(\_\_and\_< [is\\_nothrow\\_move\\_constructible](#)< \_Tp >, [is\\_nothrow\\_move\\_assignable](#)< \_Tp > >::value)
- template<typename \_Tp>  
constexpr void **swap** (\_Tp &\_\_a, \_Tp &\_\_b) noexcept([is\\_nothrow\\_move\\_assignable](#)< \_Tp > >)
- template<typename \_Tp, size\_t \_Nm>  
constexpr void **swap** (\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm]) noexcept([is\\_nothrow\\_move\\_assignable](#)< \_Tp > >)
- template<typename \_Tp, size\_t \_Nm>  
constexpr \_\_enable\_if\_t< \_\_is\_swappable< \_Tp >::value > **swap** (\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm]) noexcept(\_\_is\_nothrow\_swappable< \_Tp >::value)
- template<typename \_Tp, std::size\_t \_Nm>  
\_\_enable\_if\_t<!\_\_array\_traits< \_Tp, \_Nm >::is\_swappable::value > **swap** (array< \_Tp, \_Nm > &, array< \_Tp, \_Nm > &)=delete
- template<typename \_Tp, std::size\_t \_Nm>  
constexpr \_\_enable\_if\_t< \_\_array\_traits< \_Tp, \_Nm >::is\_swappable::value > **swap** (array< \_Tp, \_Nm > &\_\_one, array< \_Tp, \_Nm > &\_\_two) noexcept(noexcept(\_\_one.swap(\_\_two)))
- template<class \_CharT, class \_Traits>  
void **swap** ([basic\\_filebuf](#)< \_CharT, \_Traits > &\_\_x, [basic\\_filebuf](#)< \_CharT, \_Traits > &\_\_y)
- template<class \_CharT, class \_Traits>  
void **swap** ([basic\\_fstream](#)< \_CharT, \_Traits > &\_\_x, [basic\\_fstream](#)< \_CharT, \_Traits > &\_\_y)
- template<class \_CharT, class \_Traits>  
void **swap** ([basic\\_ifstream](#)< \_CharT, \_Traits > &\_\_x, [basic\\_ifstream](#)< \_CharT, \_Traits > &\_\_y)

- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap (basic_istream< _CharT, _Traits, _Allocator > &__x, basic_istream< _CharT, _Traits, _↵`  
`_Allocator > &__y)`
- `template<class _CharT, class _Traits>`  
`void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits,`  
`_Allocator > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc >`  
`&__rhs) noexcept(/*conditional */)`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator >`  
`&__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _CharT, class _Traits, class _Allocator>`  
`void swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _↵`  
`_Allocator > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `constexpr void swap (exception_ptr &__lhs, exception_ptr &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.↵`  
`swap(__ly)))`
- `template<typename _Res, typename... _Args>`  
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept`
- `template<typename _Tp, typename _Alloc>`  
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__↵`  
`y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`  
`noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(↵`  
`/*conditional */)`
- `template<typename _Res, typename... _ArgTypes>`  
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept`
- `template<typename _Tp, typename _Sequence, typename _Compare>`  
`constexpr enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::↵`  
`type swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare`  
`> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Res>`  
`void swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq >`  
`&__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional`  
`*/)`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq >`  
`&__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename... _Elements>`  
`constexpr enable_if<!__and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > & __x, tuple< _Elements... > & __y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Dp>`  
`enable_if<!__is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > & __x, unordered_set< _Value, _Hash, _Pred, _Alloc > & __y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc>`  
`constexpr void swap (vector< _Tp, _Alloc > & __x, vector< _Tp, _Alloc > & __y) noexcept(/*conditional */)`
- `void swap (thread & __x, thread & __y) noexcept`
- `template<typename _Mutex>`  
`void swap (unique_lock< _Mutex > & __x, unique_lock< _Mutex > & __y) noexcept`
- `template<typename _Tp, typename _Dp>`  
`constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > & __x, unique_ptr< _Tp, _Dp > & __y) noexcept`
- `template<typename _Tp>`  
`void swap (weak_ptr< _Tp > & __a, weak_ptr< _Tp > & __b) noexcept`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `const error_category & system_category () noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type tan (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std:: _Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp>`  
`complex< _Tp > tan (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std:: _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > & __v)`
- `constexpr float tan (float __x)`
- `constexpr long double tan (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if< __is_integer< _Tp >::value, double >::type tanh (_Tp __x)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std:: _Tanh, _Expr, _Dom >, typename _Dom::value_type > tanh (const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<typename _Tp>`  
`complex< _Tp > tanh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `constexpr float tanh (float __x)`
- `constexpr long double tanh (long double __x)`
- `void terminate () noexcept`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tgamma (_Tp __x)`
- `constexpr float tgamma (float __x)`
- `constexpr long double tgamma (long double __x)`
- `template<typename _Tp>`  
`void throw_with_nested (_Tp &&__t)`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > tie (_Elements &... __args) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp * to_address (_Tp *__ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr auto to_address (const _Ptr &__ptr) noexcept`
- `to_chars_result to_chars (char *, char *, bool, int=10)=delete`
- `constexpr to_chars_result to_chars (char *__first, char *__last, char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, signed char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, signed int __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, signed long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, signed long long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, signed short __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, unsigned char __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, unsigned int __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, unsigned long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, unsigned long long __value, int __base=10)`
- `constexpr to_chars_result to_chars (char *__first, char *__last, unsigned short __value, int __base=10)`
- `string to_string (double __val)`
- `string to_string (float __val)`
- `string to_string (int __val) noexcept`
- `string to_string (long __val) noexcept`
- `string to_string (long double __val)`
- `string to_string (long long __val)`
- `string to_string (unsigned __val) noexcept`
- `string to_string (unsigned long __val) noexcept`
- `string to_string (unsigned long long __val)`
- `wstring to_wstring (double __val)`
- `wstring to_wstring (float __val)`
- `wstring to_wstring (int __val)`
- `wstring to_wstring (long __val)`
- `wstring to_wstring (long double __val)`
- `wstring to_wstring (long long __val)`
- `wstring to_wstring (unsigned __val)`
- `wstring to_wstring (unsigned long __val)`
- `wstring to_wstring (unsigned long long __val)`
- `template<typename _CharT>`  
`_CharT tolower (_CharT __c, const locale &__loc)`

- `template<typename _CharT>`  
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`constexpr _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation>`  
`constexpr _OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`  
`constexpr _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>`  
`constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _Tp transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::enable_if<__is_integer<_Tp>::value, double>::type trunc (_Tp __x)`
- `constexpr float trunc (float __x)`
- `constexpr long double trunc (long double __x)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`int try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `template<typename... _UTypes>`  
`tuple (_UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>`  
`tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>`  
`tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>`  
`tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>`  
`tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::type`
- `bool uncaught_exception () noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp * uninitialized_construct_using_allocator (_Tp * __p, const _Alloc &__a, _Args &&... __args)`



- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __↵`  
`result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr _ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)`
- `template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr _ForwardIterator uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __↵`  
`result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr pair< _InputIterator, _ForwardIterator > uninitialized_move_n (_InputIterator __first, _Size __count,`  
`_ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)`
- `template<typename _Filter>`  
`constexpr _Filter unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate>`  
`constexpr _OIter unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`  
`_BinaryPredicate __binary_pred)`
- `ios_base & unitbuf (ios_base & __base)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵`  
`_Allocator<_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >,`  
`__iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator`  
`> >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵`  
`_Allocator<_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator)`  
`-> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<`  
`_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`



- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map< int, int >::size_type`, `_Hash`, `↵`  
`_Allocator`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash,`  
`equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↵`  
`Allocator>>>`  
**unordered\_map** (`_InputIterator`, `_InputIterator`, `typename unordered_map< int, int >::size_type={}`, `_Hash=↵`  
`Hash()`, `_Pred=_Pred()`, `_Allocator=_Allocator()`) -> `unordered_map< __iter_key_t< _InputIterator >, __iter_↵`  
`val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp > >`, `_Allocator`) -> `unordered_map< _Key, _Tp, hash< _Key`  
`>, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp > >`, `typename unordered_map< int, int >::size_type`, `↵`  
`_Allocator`) -> `unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp > >`, `typename unordered_map< int, int >::size_type`, `_Hash,`  
`_Allocator`) -> `unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_map** (`initializer_list< pair< _Key, _Tp > >`, `typename unordered_map< int, int >::size_type={}`,  
`_Hash=_Hash()`, `_Pred=_Pred()`, `_Allocator=_Allocator()`) -> `unordered_map< _Key, _Tp, _Hash, _Pred, ↵`  
`Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵`  
`Allocator<_Allocator>>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multimap< __iter_key_t< ↵`  
`_InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_↵`  
`key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵`  
`Allocator<_Allocator>>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type`, `_Allocator`) ->  
`unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<`  
`_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↵`  
`_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type`, `_Hash`, `↵`  
`_Allocator`) -> `unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash,`  
`equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↵`  
`Allocator>>>`  
**unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap< int, int >::size_type={}`, `_Hash=↵`  
`Hash()`, `_Pred=_Pred()`, `_Allocator=_Allocator()`) -> `unordered_multimap< __iter_key_t< _InputIterator >, ↵`  
`iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
**unordered\_multimap** (`initializer_list< pair< _Key, _Tp > >`, `_Allocator`) -> `unordered_multimap< _Key, _Tp,`  
`hash< _Key >, equal_to< _Key >, _Allocator >`

- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multiset`< `typename iterator_traits<_InputIterator>::value_type`, `hash`< `typename iterator_traits<_InputIterator>::value_type` >, `equal_to`< `typename iterator_traits<_InputIterator>::value_type` >, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`_InputIterator`, `_InputIterator`, `unordered_multiset`< `int` >::`size_type`, `_Allocator`) -> `unordered_multiset`< `typename iterator_traits<_InputIterator>::value_type`, `hash`< `typename iterator_traits<_InputIterator>::value_type` >, `equal_to`< `typename iterator_traits<_InputIterator>::value_type` >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`_InputIterator`, `_InputIterator`, `unordered_multiset`< `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multiset`< `typename iterator_traits<_InputIterator>::value_type`, `hash`< `typename iterator_traits<_InputIterator>::value_type` >, `equal_to`< `typename iterator_traits<_InputIterator>::value_type` >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`_InputIterator`, `_InputIterator`, `unordered_multiset`< `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multiset`< `typename iterator_traits<_InputIterator>::value_type`, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`initializer_list`< `_Tp` >, `_Allocator`) -> `unordered_multiset`< `_Tp`, `hash`< `_Tp` >, `equal_to`< `_Tp` >, `_Allocator` >
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`initializer_list`< `_Tp` >, `unordered_multiset`< `int` >::`size_type`, `_Allocator`) -> `unordered_multiset`< `_Tp`, `hash`< `_Tp` >, `equal_to`< `_Tp` >, `_Allocator` >
- `template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`initializer_list`< `_Tp` >, `unordered_multiset`< `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multiset`< `_Tp`, `_Hash`, `equal_to`< `_Tp` >, `_Allocator` >
- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**unordered\_multiset** (`initializer_list`< `_Tp` >, `unordered_multiset`< `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multiset`< `_Tp`, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`

- unordered\_set** (*\_InputIterator*, *\_InputIterator*, *\_Allocator*) -> *unordered\_set*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type*, *hash*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type* >, *equal\_to*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type* >, *\_Allocator* >
- *template*<*typename* *\_InputIterator*, *typename* *\_Allocator*, *typename* = *\_RequireInputIter*<*\_InputIterator*>, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*\_InputIterator*, *\_InputIterator*, *unordered\_set*< *int* >::*size\_type*, *\_Allocator*) -> *unordered\_set*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type*, *hash*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type* >, *equal\_to*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type* >, *\_Allocator* >
  - *template*<*typename* *\_InputIterator*, *typename* *\_Hash*, *typename* *\_Allocator*, *typename* = *\_RequireInputIter*<*\_InputIterator*>, *typename* = *\_RequireNotAllocatorOrIntegral*<*\_Hash*>, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*\_InputIterator*, *\_InputIterator*, *unordered\_set*< *int* >::*size\_type*, *\_Hash*, *\_Allocator*) -> *unordered\_set*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type*, *\_Hash*, *equal\_to*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type* >, *\_Allocator* >
  - *template*<*typename* *\_InputIterator*, *typename* *\_Hash* = *hash*<*typename iterator\_traits*<*\_InputIterator*>::*value\_type*>, *typename* *\_Pred* = *equal\_to*<*typename iterator\_traits*<*\_InputIterator*>::*value\_type*>, *typename* *\_Allocator* = *allocator*<*typename iterator\_traits*<*\_InputIterator*>::*value\_type*>, *typename* = *\_RequireInputIter*<*\_InputIterator*>, *typename* = *\_RequireNotAllocatorOrIntegral*<*\_Hash*>, *typename* = *\_RequireNotAllocator*<*\_Pred*>, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*\_InputIterator*, *\_InputIterator*, *unordered\_set*< *int* >::*size\_type*={}, *\_Hash*=*\_Hash*(), *\_Pred*=*\_Pred*(), *\_Allocator*=*\_Allocator*()) -> *unordered\_set*< *typename iterator\_traits*< *\_InputIterator* >::*value\_type*, *\_Hash*, *\_Pred*, *\_Allocator* >
  - *template*<*typename* *\_Tp*, *typename* *\_Allocator*, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*initializer\_list*< *\_Tp* >, *\_Allocator*) -> *unordered\_set*< *\_Tp*, *hash*< *\_Tp* >, *equal\_to*< *\_Tp* >, *\_Allocator* >
  - *template*<*typename* *\_Tp*, *typename* *\_Allocator*, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*initializer\_list*< *\_Tp* >, *unordered\_set*< *int* >::*size\_type*, *\_Allocator*) -> *unordered\_set*< *\_Tp*, *hash*< *\_Tp* >, *equal\_to*< *\_Tp* >, *\_Allocator* >
  - *template*<*typename* *\_Tp*, *typename* *\_Hash*, *typename* *\_Allocator*, *typename* = *\_RequireNotAllocatorOrIntegral*<*\_Hash*>, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*initializer\_list*< *\_Tp* >, *unordered\_set*< *int* >::*size\_type*, *\_Hash*, *\_Allocator*) -> *unordered\_set*< *\_Tp*, *\_Hash*, *equal\_to*< *\_Tp* >, *\_Allocator* >
  - *template*<*typename* *\_Tp*, *typename* *\_Hash* = *hash*<*\_Tp*>, *typename* *\_Pred* = *equal\_to*<*\_Tp*>, *typename* *\_Allocator* = *allocator*<*\_Tp*>, *typename* = *\_RequireNotAllocatorOrIntegral*<*\_Hash*>, *typename* = *\_RequireNotAllocator*<*\_Pred*>, *typename* = *\_RequireAllocator*<*\_Allocator*>>  
**unordered\_set** (*initializer\_list*< *\_Tp* >, *unordered\_set*< *int* >::*size\_type*={}, *\_Hash*=*\_Hash*(), *\_Pred*=*\_Pred*(), *\_Allocator*=*\_Allocator*()) -> *unordered\_set*< *\_Tp*, *\_Hash*, *\_Pred*, *\_Allocator* >
  - *template*<*typename* *\_Filter*, *typename* *\_Tp* = *typename iterator\_traits*<*\_Filter*>::*value\_type*>  
**constexpr** *\_Filter* **upper\_bound** (*\_Filter*, *\_Filter*, *const* *\_Tp* &)
  - *template*<*typename* *\_Filter*, *typename* *\_Tp* = *typename iterator\_traits*<*\_Filter*>::*value\_type*, *typename* *\_Compare*>  
**constexpr** *\_Filter* **upper\_bound** (*\_Filter*, *\_Filter*, *const* *\_Tp* &, *\_Compare*)
  - *template*<*typename* *\_ForwardIterator*, *typename* *\_Tp*>  
**constexpr** *\_ForwardIterator* **upper\_bound** (*\_ForwardIterator* \_\_first, *\_ForwardIterator* \_\_last, *const* *\_Tp* & \_\_val)
  - *template*<*typename* *\_ForwardIterator*, *typename* *\_Tp*, *typename* *\_Compare*>  
**constexpr** *\_ForwardIterator* **upper\_bound** (*\_ForwardIterator* \_\_first, *\_ForwardIterator* \_\_last, *const* *\_Tp* & \_\_val, *\_Compare* \_\_comp)
  - *ios\_base* & **uppercase** (*ios\_base* & \_\_base)
  - *template*<*typename* *\_Facet*>  
**const** *\_Facet* & **use\_facet** (*const* *locale* & \_\_loc)
  - *template* *const* **codecvt**< *char*, *char*, *mbstate\_t* > & **use\_facet**< **codecvt**< *char*, *char*, *mbstate\_t* > > (*const* *locale* &)
  - *template* *const* **codecvt**< *wchar\_t*, *char*, *mbstate\_t* > & **use\_facet**< **codecvt**< *wchar\_t*, *char*, *mbstate\_t* > > (*const* *locale* &)
  - *template*<*\_Std\_pair* *\_Tp*, *typename* *\_Alloc*>  
**constexpr** *auto* **uses\_allocator\_construction\_args** (*const* *\_Alloc* &) *noexcept*

- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &, _Up &&, _Vp &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`requires (! _Std_pair<_Tp>)`  
`constexpr auto uses_allocator_construction_args (const _Alloc &__a, _Args &&... __args) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>`  
`constexpr auto uses_allocator_construction_args (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >`
- `wchar_t * wcschr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsbrk (wchar_t * __s1, const wchar_t * __s2)`
- `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t * __s1, const wchar_t * __s2)`
- `template<typename _Tp>`  
`weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >`
- `wchar_t * wmemchr (wchar_t * __p, wchar_t __c, size_t __n)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > & __is)`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator& (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator| (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator^ (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > & __x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb > & __x)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > & __x, const complex< _Tp > & __y)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > & __x, const _Tp & __y)`

- `template<typename _Tp>`  
`constexpr complex<_Tp> operator+ (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator- (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator- (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator- (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator* (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator* (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator* (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator/ (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator/ (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> operator/ (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr bool operator== (const complex<_Tp> &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr bool operator== (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream<_CharT, _Traits> & operator>> (basic_istream<_CharT, _Traits> &__in, _CharT &__c)`
- `template<class _Traits>`  
`basic_istream<char, _Traits> & operator>> (basic_istream<char, _Traits> &__in, unsigned char &__c)`
- `template<class _Traits>`  
`basic_istream<char, _Traits> & operator>> (basic_istream<char, _Traits> &__in, signed char &__c)`
- `template<typename _CharT, typename _Traits, size_t _Num>`  
`basic_istream<_CharT, _Traits> & operator>> (basic_istream<_CharT, _Traits> &__in, _CharT(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream<char, _Traits> & operator>> (basic_istream<char, _Traits> &__in, unsigned char(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream<char, _Traits> & operator>> (basic_istream<char, _Traits> &__in, signed char(&__s)[_Num])`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, const _CharT * __s)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<<< (basic_ostream< _CharT, _Traits > &__out, const char * __s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, const char * __s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, const signed char * __s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & operator<<< (basic_ostream< char, _Traits > &__out, const unsigned char * __s)`

### Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits>`  
`bool regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits>`  
`bool regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search ( _Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`



- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search ( _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits>`  
`bool regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex<`  
`_Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_`  
`_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _Sa>`  
`_Out_iter regex_replace ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _`  
`_Rx_traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter regex_replace ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _`  
`Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s,`  
`const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const`  
`basic_regex< _Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_`  
`traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type>`  
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits >`  
`& __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)`
- `template<typename _Tp>`  
`bool atomic_is_lock_free (const shared_ptr< _Tp > * __p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > * __p, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > * __p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic_load_explicit (const __shared_ptr< _Tp, _Lp > * __p, memory_order)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp>`  
`void atomic\_store\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r, memory\_order)`
- `template<typename _Tp>`  
`void atomic\_store (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void atomic\_store\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void atomic\_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > atomic\_exchange\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r, memory\_order)`
- `template<typename _Tp>`  
`shared\_ptr< _Tp > atomic\_exchange (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_exchange\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > atomic\_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`bool atomic\_compare\_exchange\_strong\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > *__v, shared\_ptr< _Tp > __w, memory\_order, memory\_order)`
- `template<typename _Tp>`  
`bool atomic\_compare\_exchange\_strong (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > *__v, shared\_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool atomic\_compare\_exchange\_weak\_explicit (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > *__v, shared\_ptr< _Tp > __w, memory\_order __success, memory\_order __failure)`
- `template<typename _Tp>`  
`bool atomic\_compare\_exchange\_weak (shared\_ptr< _Tp > *__p, shared\_ptr< _Tp > *__v, shared\_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_strong\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory\_order, memory\_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_weak\_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory\_order __success, memory\_order __failure)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool atomic\_compare\_exchange\_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`



- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (const pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<size_t _Int, class _Tp1, class _Tp2>`  
`constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (const pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept`

## Variables

- `template<typename _ValT, typename _Tp>`  
`constexpr bool __can_use_memchr_for_find`
- `const _Lock_policy __default_lock_policy`
- `template<typename _Tp>`  
`constexpr auto __denorm_min_v`
- `template<typename _Tp>`  
`constexpr auto __digits10_v`
- `template<typename _Tp>`  
`constexpr auto __digits_v`
- `template<typename _Tp>`  
`constexpr auto __epsilon_v`
- `template<typename _Tp>`  
`constexpr auto __finite_max_v`
- `template<typename _Tp>`  
`constexpr auto __finite_min_v`
- `template<template< typename > class _Trait, typename _Tp>`  
`constexpr bool __has_iec559_behavior_v`
- `template<typename _Tp>`  
`constexpr bool __has_iec559_storage_format_v`
- `template<typename _Tp>`  
`constexpr auto __infinity_v`
- `static ios_base::Init __ioinit`
- `template<typename >`  
`constexpr bool __is_constant_wrapper_v`

- `template<auto __cw, typename _Fn>`  
`constexpr bool __is_constant_wrapper_v< constant_wrapper< __cw, _Fn > >`
- `template<typename _Tp, typename = void>`  
`constexpr bool __is_hash_enabled_for`
- `template<typename _Tp>`  
`constexpr bool __is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp >()(declval< _Tp >()))> >`
- `template<typename>`  
`constexpr bool __is_in_place_index_v`
- `template<size_t _Nm>`  
`constexpr bool __is_in_place_index_v< in_place_index_t< _Nm > >`
- `template<typename>`  
`constexpr bool __is_in_place_type_v`
- `template<typename _Tp>`  
`constexpr bool __is_in_place_type_v< in_place_type_t< _Tp > >`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool __is_nothrow_new_constructible`
- `template<typename _Tp>`  
`constexpr bool __is_pair`
- `template<typename _Tp, typename _Up>`  
`constexpr bool __is_pair< pair< _Tp, _Up > >`
- `template<typename _Tp>`  
`const bool __is_rand_dist_float_v`
- `template<template< typename > class>`  
`constexpr bool __is_std_op_template`
- `template<> constexpr bool __is_std_op_template< std::bit_and >`
- `template<> constexpr bool __is_std_op_template< std::bit_not >`
- `template<> constexpr bool __is_std_op_template< std::bit_or >`
- `template<> constexpr bool __is_std_op_template< std::bit_xor >`
- `template<> constexpr bool __is_std_op_template< std::divides >`
- `template<> constexpr bool __is_std_op_template< std::equal_to >`
- `template<> constexpr bool __is_std_op_template< std::greater >`
- `template<> constexpr bool __is_std_op_template< std::greater_equal >`
- `template<> constexpr bool __is_std_op_template< std::less >`
- `template<> constexpr bool __is_std_op_template< std::less_equal >`
- `template<> constexpr bool __is_std_op_template< std::logical_and >`
- `template<> constexpr bool __is_std_op_template< std::logical_not >`
- `template<> constexpr bool __is_std_op_template< std::logical_or >`
- `template<> constexpr bool __is_std_op_template< std::minus >`
- `template<> constexpr bool __is_std_op_template< std::modulus >`
- `template<> constexpr bool __is_std_op_template< std::multiplies >`
- `template<> constexpr bool __is_std_op_template< std::negate >`
- `template<> constexpr bool __is_std_op_template< std::not_equal_to >`
- `template<> constexpr bool __is_std_op_template< std::plus >`
- `template<typename _Fn>`  
`constexpr bool __is_std_op_wrapper`
- `template<template< typename > class _Ft, typename _Tp>`  
`constexpr bool __is_std_op_wrapper< _Ft< _Tp > >`
- `template<typename _Tp>`  
`constexpr bool __is_submdspan_mapping_result`
- `template<typename _Mapping>`  
`constexpr bool __is_submdspan_mapping_result< submdspan_mapping_result< _Mapping > >`

- `template<typename _It, typename _Sent, ranges::subrange_kind _Kind>`  
`constexpr bool __is_tuple_like_v< ranges::subrange< _It, _Sent, _Kind > >`
- `template<typename _Tp>`  
`constexpr auto __max_digits10_v`
- `template<typename _Tp>`  
`constexpr auto __max_exponent10_v`
- `template<typename _Tp>`  
`constexpr auto __max_exponent_v`
- `template<typename _Tp>`  
`constexpr auto __min_exponent10_v`
- `template<typename _Tp>`  
`constexpr auto __min_exponent_v`
- `template<typename _Tp>`  
`constexpr auto __norm_min_v`
- `template<typename _Tp>`  
`constexpr auto __quiet_NaN_v`
- `template<typename _Tp>`  
`constexpr auto __radix_v`
- `template<typename _Tp>`  
`constexpr auto __reciprocal_overflow_threshold_v`
- `template<typename _Tp>`  
`constexpr auto __round_error_v`
- `template<typename _Tp>`  
`constexpr auto __signaling_NaN_v`
- `template<typename _CharT>`  
`locale::id __timepunct< _CharT >::id`
- `template<typename _CharT>`  
`const _CharT * __timepunct_cache< _CharT >::S_timezones [14]`
- `template<> const char * __timepunct_cache< char >::S_timezones [14]`
- `template<> const wchar_t * __timepunct_cache< wchar_t >::S_timezones [14]`
- `template<template< typename > class _Trait, typename _Tp>`  
`constexpr bool __value_exists_v`
- `constexpr adopt_lock_t adopt_lock`
- `template<size_t _Len, typename... _Types>`  
`const size_t aligned_union< _Len, _Types... >::alignment_value`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`const basic_string< _CharT, _Traits, _Alloc >::size_type basic_string< _CharT, _Traits, _Alloc >::npos`
- `template<typename _InternT, typename _ExternT, typename _StateT>`  
`locale::id codecvt< _InternT, _ExternT, _StateT >::id`
- `template<typename _InternT, typename _ExternT>`  
`locale::id codecvt< _InternT, _ExternT, encoding_state >::id`
- `template<typename _CharT>`  
`locale::id collate< _CharT >::id`
- `constexpr __compare::Partial_fallback compare_partial_order_fallback`
- `constexpr __compare::Strong_fallback compare_strong_order_fallback`
- `constexpr __compare::Weak_fallback compare_weak_order_fallback`
- `template<typename _CharT>`  
`locale::id ctype< _CharT >::id`
- `template<auto _Xv>`  
`constexpr auto cw`
- `constexpr default_sentinel_t default_sentinel`
- `constexpr defer_lock_t defer_lock`

- constexpr [destroying\\_delete\\_t](#) [destroying\\_delete](#)
- template<typename \_Sent, typename \_Iter>  
constexpr bool **disable\_sized\_sentinel\_for**
- template<typename \_Iterator1, typename \_Iterator2>  
constexpr bool **disable\_sized\_sentinel\_for**< [move\\_iterator](#)< \_Iterator1 >, [move\\_iterator](#)< \_Iterator2 > >
- template<typename \_Iterator1, typename \_Iterator2>  
constexpr bool **disable\_sized\_sentinel\_for**< [reverse\\_iterator](#)< \_Iterator1 >, [reverse\\_iterator](#)< \_Iterator2 > >
- constexpr full\_extent\_t **full\_extent**
- constexpr \_Swallow\_assign [ignore](#)
- constexpr in\_place\_t **in\_place**
- template<size\_t \_Idx>  
constexpr in\_place\_index\_t< \_Idx > **in\_place\_index**
- template<typename \_Tp>  
constexpr in\_place\_type\_t< \_Tp > **in\_place\_type**
- template<typename \_Tp>  
constexpr bool **is\_bind\_expression\_v**
- template<typename \_Tp>  
constexpr bool **is\_error\_code\_enum\_v**
- template<typename \_Tp>  
constexpr bool **is\_error\_condition\_enum\_v**
- template<typename \_Tp>  
constexpr int **is\_placeholder\_v**
- constexpr [memory\\_order](#) [memory\\_order\\_acq\\_rel](#)
- constexpr [memory\\_order](#) [memory\\_order\\_acquire](#)
- constexpr [memory\\_order](#) [memory\\_order\\_consume](#)
- constexpr [memory\\_order](#) [memory\\_order\\_relaxed](#)
- constexpr [memory\\_order](#) [memory\\_order\\_release](#)
- constexpr [memory\\_order](#) [memory\\_order\\_seq\\_cst](#)
- template<typename \_CharT>  
[locale::id](#) **messages**< \_CharT >::id
- template<typename \_CharT, typename \_InIter>  
[locale::id](#) **money\_get**< \_CharT, \_InIter >::id
- template<typename \_CharT, typename \_OutIter>  
[locale::id](#) **money\_put**< \_CharT, \_OutIter >::id
- template<typename \_CharT, bool \_Intl>  
[locale::id](#) **money\_punct**< \_CharT, \_Intl >::id
- template<typename \_CharT, bool \_Intl>  
const bool **money\_punct**< \_CharT, \_Intl >::intl
- template<typename \_CharT, bool \_Intl>  
const bool **money\_punct\_byname**< \_CharT, \_Intl >::intl
- const nothrow\_t **nothrow**
- template<typename \_CharT, typename \_InIter>  
[locale::id](#) **num\_get**< \_CharT, \_InIter >::id
- template<typename \_CharT, typename \_OutIter>  
[locale::id](#) **num\_put**< \_CharT, \_OutIter >::id
- template<typename \_CharT>  
[locale::id](#) **num\_punct**< \_CharT >::id
- constexpr \_\_compare::Partial\_order **partial\_order**
- constexpr [piecewise\\_construct\\_t](#) [piecewise\\_construct](#)
- template<typename \_R1, typename \_R2>  
constexpr bool **ratio\_equal\_v**

- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_greater_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_equal_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_less_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio_not_equal_v`
- `constexpr sorted_equivalent_t sorted_equivalent`
- `constexpr sorted_unique_t sorted_unique`
- `constexpr __compare::__Strong_order strong_order`
- `template<typename _CharT, typename _InIter>`  
`locale::id time_get< _CharT, _InIter >::id`
- `template<typename _CharT, typename _OutIter>`  
`locale::id time_put< _CharT, _OutIter >::id`
- `constexpr try\_to\_lock\_t try_to_lock`
- `template<typename _Tp>`  
`constexpr size_t tuple_size_v`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t tuple_size_v< array< _Tp, _Nm > >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t tuple_size_v< const array< _Tp, _Nm > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t tuple_size_v< const pair< _Tp1, _Tp2 > >`
- `template<typename... _Types>`  
`constexpr size_t tuple_size_v< const tuple< _Types... > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t tuple_size_v< pair< _Tp1, _Tp2 > >`
- `template<typename... _Types>`  
`constexpr size_t tuple_size_v< tuple< _Types... > >`
- `template<typename _Tp, typename _Up>`  
`constexpr strong_ordering type\_order\_v`
- `constexpr unreachable_sentinel_t unreachable_sentinel`
- `constexpr __compare::__Weak_order weak_order`

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- `istream cin`
- `ostream cout`
- `ostream cerr`
- `ostream clog`
- `wistream wcin`
- `wostream wcout`
- `wostream wcerr`
- `wostream wclog`
- `template<typename... _Tp>`  
`using common\_reference\_t`

- `template<typename _Tp, typename _Up>`  
`constexpr bool is\_layout\_compatible\_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool is\_pointer\_interconvertible\_base\_of\_v`
- `template<typename _S1, typename _S2, typename _M1, typename _M2>`  
`constexpr bool is\_corresponding\_member (_M1 _S1::* __m1, _M2 _S2::* __m2) noexcept`
- `template<typename _Tp, typename _Mem>`  
`constexpr bool is\_pointer\_interconvertible\_with\_class (_Mem _Tp::* __mp) noexcept`

#### 4.10.1 Detailed Description

ISO C++ entities toplevel namespace is std.

#### 4.10.2 Typedef Documentation

##### **`__ptr_rebind`**

```
template<typename _Ptr, typename _Tp>
using std::__ptr_rebind
Convenience alias for rebinding pointers.
```

##### **`__umap_traits`**

```
template<bool _Cache>
using std::__umap_traits
Base types for unordered_map.
```

##### **`__ummap_traits`**

```
template<bool _Cache>
using std::__ummap_traits
Base types for unordered_multimap.
```

##### **`__umset_traits`**

```
template<bool _Cache>
using std::__umset_traits
Base types for unordered_multiset.
```

##### **`__uset_traits`**

```
template<bool _Cache>
using std::__uset_traits
Base types for unordered_set.
```

##### **`compare_three_way_result_t`**

```
template<typename _Tp, typename _Up = _Tp>
using std::compare_three_way_result_t
[cmp.result], result of three-way comparison
```

##### **`index_sequence`**

```
template<size_t... _Idx>
using std::index_sequence
Alias template index_sequence.
```

**index\_sequence\_for**

```
template<typename... _Types>
using std::index_sequence_for
Alias template index_sequence_for.
```

**iter\_rvalue\_reference\_t**

```
template<__detail::__dereferenceable _Tp>
using std::iter_rvalue_reference_t
The result type of ranges::iter_move(std::declval<_Tp>()).
```

**make\_index\_sequence**

```
template<size_t _Num>
using std::make_index_sequence
Alias template make_index_sequence.
```

**make\_integer\_sequence**

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence
Alias template make_integer_sequence.
```

**new\_handler**

```
typedef void(* std::new_handler) ()
If you write your own error handler to be called by new, it must be of this type.
```

**projected**

```
template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
using std::projected
[projected], projected
```

**streamoff**

```
typedef long long std::streamoff
Type used by fpos, char_traits<char>, and char_traits<wchar_t>.
In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.
```

**streampos**

```
typedef fpos<mbstate_t> std::streampos
File position for char streams.
```

**streamsize**

```
typedef ptrdiff_t std::streamsize
Integral type for I/O operation counts and buffer sizes.
```

**u16streampos**

```
typedef fpos<mbstate_t> std::u16streampos
File position for char16_t streams.
```

**u32streampos**

```
typedef fpos<mbstate_t> std::u32streampos
```

File position for char32\_t streams.

**wstreampos**

```
typedef fpos<mbstate_t> std::wstreampos
```

File position for wchar\_t streams.

**4.10.3 Enumeration Type Documentation****chars\_format**

```
enum class std::chars_format [strong]
```

floating-point format for primitive numerical conversion

**float\_denorm\_style**

```
enum std::float_denorm_style
```

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the std::numeric\_limits class.

**Enumerator**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| denorm_indeterminate | Indeterminate at compile time whether denormalized values are allowed. |
| denorm_absent        | The type does not allow denormalized values.                           |
| denorm_present       | The type allows denormalized values.                                   |

**float\_round\_style**

```
enum std::float_round_style
```

Describes the rounding style for floating-point types.

This is used in the std::numeric\_limits class.

**Enumerator**

|                           |                                     |
|---------------------------|-------------------------------------|
| round_indeterminate       | Intermediate.                       |
| round_toward_zero         | To zero.                            |
| round_to_nearest          | To the nearest representable value. |
| round_toward_infinity     | To infinity.                        |
| round_toward_neg_infinity | To negative infinity.               |

**io\_errc**

```
enum class std::io_errc [strong]
```

I/O error code.



#### 4.10.4 Function Documentation

##### `__find_if_not()`

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

Provided for `stable_partition` to use.

Referenced by [find\\_if\\_not\(\)](#).

##### `__find_if_not_n()`

```
template<typename _InputIterator, typename _Predicate, typename _Distance>
_InputIterator std::__find_if_not_n (
 _InputIterator __first,
 _Distance & __len,
 _Predicate __pred) [constexpr]
```

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Referenced by [\\_\\_stable\\_partition\\_adaptive\(\)](#).

##### `__gcd()`

```
template<typename _EuclideanRingElement>
_EuclideanRingElement std::__gcd (
 _EuclideanRingElement __m,
 _EuclideanRingElement __n) [constexpr]
```

This is a helper function for the rotate algorithm specialized on RAs. It returns the greatest common divisor of two integer values.

##### `__gen_two_uniform_ints()`

```
template<typename _IntType, typename _UniformRandomBitGenerator>
pair< _IntType, _IntType > std::__gen_two_uniform_ints (
 _IntType __b0,
 _IntType __b1,
 _UniformRandomBitGenerator && __g)
```

Generate two uniformly distributed integers using a single distribution invocation.

##### Parameters

|                   |                                            |
|-------------------|--------------------------------------------|
| <code>__b0</code> | The upper bound for the first integer.     |
| <code>__b1</code> | The upper bound for the second integer.    |
| <code>__g</code>  | A <code>UniformRandomBitGenerator</code> . |

##### Returns

A pair (i, j) with i and j uniformly distributed over [0, \_\_b0) and [0, \_\_b1), respectively.

Requires: `__b0 * __b1 <= __g.max() - __g.min()`.

Using `uniform_int_distribution` with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since `uniform_int_distribution` does not store leftover randomness between invocations.

If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.

References [make\\_pair\(\)](#).

Referenced by [\\_\\_sample\(\)](#), and [shuffle\(\)](#).

### `__inplace_stable_sort()`

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

This is a helper function for the stable sorting routines.

References [\\_\\_inplace\\_stable\\_sort\(\)](#), and [\\_\\_merge\\_without\\_buffer\(\)](#).

Referenced by [\\_\\_inplace\\_stable\\_sort\(\)](#).

### `__lg()`

```
template<typename _Tp>
_Tp std::__lg (
 _Tp __n) [inline], [constexpr]
```

This is a helper function for the sort routines and for `random.tcc`.

Referenced by [nth\\_element\(\)](#), [nth\\_element\(\)](#), [std::independent\\_bits\\_engine<\\_RandomNumberEngine, \\_\\_w, \\_UIntType >::operator\(\)\(\)](#), and [std::linear\\_congruential\\_engine<\\_UIntType, \\_\\_a, \\_\\_c, \\_\\_m >::seed\(\)](#).

### `__merge_adaptive()`

```
template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>
void std::__merge_adaptive (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 _Distance __len1,
 _Distance __len2,
 _Pointer __buffer,
 _Compare __comp)
```

This is a helper function for the merge routines.

References [\\_\\_move\\_merge\\_adaptive\(\)](#), and [\\_\\_move\\_merge\\_adaptive\\_backward\(\)](#).

### `__merge_without_buffer()`

```
template<typename _BidirectionalIterator, typename _Distance, typename _Compare>
void std::__merge_without_buffer (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 _Distance __len1,
 _Distance __len2,
 _Compare __comp) [constexpr]
```

This is a helper function for the merge routines.

References [\\_\\_merge\\_without\\_buffer\(\)](#), [advance\(\)](#), and [distance\(\)](#).

Referenced by [\\_\\_inplace\\_stable\\_sort\(\)](#), and [\\_\\_merge\\_without\\_buffer\(\)](#).

### **`__move_median_to_first()`**

```
template<typename _Iterator, typename _Compare>
void std::__move_median_to_first (
 _Iterator __result,
 _Iterator __a,
 _Iterator __b,
 _Iterator __c,
 _Compare __comp) [constexpr]
```

Swaps the median value of \*\_\_a, \*\_\_b and \*\_\_c under \_\_comp to \*\_\_result.

### **`__move_merge()`**

```
template<typename _InputIterator, typename _OutputIterator, typename _Compare>
_OutputIterator std::__move_merge (
 _InputIterator __first1,
 _InputIterator __last1,
 _InputIterator __first2,
 _InputIterator __last2,
 _OutputIterator __result,
 _Compare __comp)
```

This is a helper function for the `__merge_sort_loop` routines.

### **`__move_merge_adaptive()`**

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
void std::__move_merge_adaptive (
 _InputIterator1 __first1,
 _InputIterator1 __last1,
 _InputIterator2 __first2,
 _InputIterator2 __last2,
 _OutputIterator __result,
 _Compare __comp)
```

This is a helper function for the `__merge_adaptive` routines.

Referenced by [\\_\\_merge\\_adaptive\(\)](#).

### **`__move_merge_adaptive_backward()`**

```
template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>
void std::__move_merge_adaptive_backward (
 _BidirectionalIterator1 __first1,
 _BidirectionalIterator1 __last1,
 _BidirectionalIterator2 __first2,
 _BidirectionalIterator2 __last2,
 _BidirectionalIterator3 __result,
 _Compare __comp)
```

This is a helper function for the `__merge_adaptive` routines.

Referenced by [\\_\\_merge\\_adaptive\(\)](#).

**\_\_partition()** [1/2]

```
template<typename _BidirectionalIterator, typename _Predicate>
_BidirectionalIterator std::__partition (
 _BidirectionalIterator __first,
 _BidirectionalIterator __last,
 _Predicate __pred,
 bidirectional_iterator_tag) [constexpr]
```

This is a helper function...

**\_\_partition()** [2/2]

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

This is a helper function...

Referenced by [partition\(\)](#).

**\_\_reverse()** [1/2]

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

This is an uglified reverse(\_BidirectionalIterator, \_BidirectionalIterator) overloaded for bidirectional iterators.

Referenced by [\\_\\_rotate\(\)](#), and [reverse\(\)](#).

**\_\_reverse()** [2/2]

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

This is an uglified reverse(\_BidirectionalIterator, \_BidirectionalIterator) overloaded for random access iterators.

**\_\_rotate()** [1/3]

```
template<typename _BidirectionalIterator>
_BidirectionalIterator std::__rotate (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 bidirectional_iterator_tag) [constexpr]
```

This is a helper function for the rotate algorithm.

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

**\_\_rotate()** [2/3]

```
template<typename _ForwardIterator>
_FowardIterator std::__rotate (
```

```

 _ForwardIterator __first,
 _ForwardIterator __middle,
 _ForwardIterator __last,
 forward_iterator_tag) [constexpr]

```

This is a helper function for the rotate algorithm.

Referenced by [rotate\(\)](#).

#### **\_\_rotate()** [3/3]

```

template<typename _RandomAccessIterator>
_RandomAccessIterator std::__rotate (
 _RandomAccessIterator __first,
 _RandomAccessIterator __middle,
 _RandomAccessIterator __last,
 random_access_iterator_tag) [constexpr]

```

This is a helper function for the rotate algorithm.

#### **\_\_rotate\_adaptive()**

```

template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>
_BidirectionalIterator1 std::__rotate_adaptive (
 _BidirectionalIterator1 __first,
 _BidirectionalIterator1 __middle,
 _BidirectionalIterator1 __last,
 _Distance __len1,
 _Distance __len2,
 _BidirectionalIterator2 __buffer,
 _Distance __buffer_size)

```

This is a helper function for the merge routines.

#### **\_\_sample()** [1/2]

```

template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size,
typename _UniformRandomBitGenerator>
_OutputIterator std::__sample (
 _ForwardIterator __first,
 _ForwardIterator __last,
 forward_iterator_tag ,
 _OutputIterator __out,
 _Cat ,
 _Size __n,
 _UniformRandomBitGenerator && __g)

```

Selection sampling algorithm.

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

#### **\_\_sample()** [2/2]

```

template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _↵
UniformRandomBitGenerator>
_RandomAccessIterator std::__sample (
 _InputIterator __first,
 _InputIterator __last,
 input_iterator_tag ,
 _RandomAccessIterator __out,
 random_access_iterator_tag ,

```

```

 _Size __n,
 _UniformRandomBitGenerator && __g)

```

Reservoir sampling algorithm.

### **\_\_search\_n\_aux()** [1/2]

```

template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
_FForwardIterator std::__search_n_aux (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Integer __count,
 _UnaryPredicate __unary_pred,
 std::forward_iterator_tag) [constexpr]

```

This is an helper function for `search_n` overloaded for forward iterators.

### **\_\_search\_n\_aux()** [2/2]

```

template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>
_RandomAccessIter std::__search_n_aux (
 _RandomAccessIter __first,
 _RandomAccessIter __last,
 _Integer __count,
 _UnaryPredicate __unary_pred,
 std::random_access_iterator_tag) [constexpr]

```

This is an helper function for `search_n` overloaded for random access iterators.

### **\_\_stable\_partition\_adaptive()**

```

template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>
_FForwardIterator std::__stable_partition_adaptive (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Predicate __pred,
 _Distance __len,
 _Pointer __buffer,
 _Distance __buffer_size) [constexpr]

```

This is a helper function... Requires `__first != __last` and `!__pred(*__first)` and `__len == distance(__first, __last)`.

`!__pred(*__first)` allows us to guarantee that we don't move-assign an element onto itself.

References [\\_\\_find\\_if\\_not\\_n\(\)](#), [\\_\\_stable\\_partition\\_adaptive\(\)](#), and [advance\(\)](#).

Referenced by [\\_\\_stable\\_partition\\_adaptive\(\)](#).

### **\_Construct()**

```

template<typename _Tp, typename... _Args>
void std::_Construct (
 _Tp * __p,
 _Args &&... __args) [inline], [constexpr]

```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

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

Referenced by [std::allocator\\_traits< \\_OuterAlloc >::construct\(\)](#), [std::allocator\\_traits< allocator< \\_Tp > >::construct\(\)](#), and [std::allocator\\_traits< allocator< void > >::construct\(\)](#).

### **\_Destroy()** [1/2]

```

template<typename _ForwardIterator>

```

```
void std::_Destroy (
 _ForwardIterator __first,
 _ForwardIterator __last) [inline], [constexpr]
```

Destroy a range of objects. If the value\_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

References [\\_Destroy\(\)](#), and [addressof\(\)](#).

Referenced by [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::~~vector\(\)](#), [\\_Destroy\(\)](#), [\\_Destroy\\_n\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_std::deque<\\_Tp, \\_Alloc>::\\_M\\_range\\_initialize\(\)](#), [std::allocator\\_traits<\\_OuterAlloc>::destroy\(\)](#), [std::allocator\\_traits<allocator<void>>::destroy\(\)](#), [std::vector<\\_Tp, \\_Alloc>::operator=\(\)](#), and [std::vector<\\_Tp, \\_Alloc>::reserve\(\)](#).

### **[\\_Destroy\(\)](#) [2/2]**

```
template<typename _Tp>
void std::_Destroy (
 _Tp * __pointer) [inline], [constexpr]
```

Destroy the object pointed to by a pointer type.

### **[\\_Destroy\\_n\(\)](#)**

```
template<typename _ForwardIterator, typename _Size>
_FowardIterator std::_Destroy_n (
 _ForwardIterator __first,
 _Size __count) [inline], [constexpr]
```

Destroy a range of objects. If the value\_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

References [\\_Destroy\(\)](#), [addressof\(\)](#), and [advance\(\)](#).

### **[abs\(\)](#)**

```
template<typename _Tp>
_Tp std::abs (
 const complex<_Tp> & __z) [inline]
```

Return magnitude of z.

References [abs\(\)](#), and [arg\(\)](#).

### **[acosh\(\)](#)**

```
float std::acosh (
 float __x) [constexpr]
```

Additional overloads.

### **[advance\(\)](#)**

```
template<typename _InputIterator, typename _Distance>
void std::advance (
 _InputIterator & __i,
 _Distance __n) [inline], [constexpr]
```

A generalization of pointer arithmetic.

**Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__i</code> | An input iterator.                                     |
| <code>__n</code> | The <i>delta</i> by which to change <code>__i</code> . |

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Referenced by [\\_\\_merge\\_without\\_buffer\(\)](#), [\\_\\_stable\\_partition\\_adaptive\(\)](#), [\\_Destroy\\_n\(\)](#), [std::deque<\\_Tp, \\_Alloc>::\\_M\\_range\\_initialize\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc>::get\\_child\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc>::get\\_child\(\)](#), and [partition\\_point\(\)](#).

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

```
template<typename _Container>
auto std::begin (
 _Container & __cont) -> decltype(__cont.begin()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++11

Referenced by [cbegin\(\)](#), [std::vector<\\_Tp, \\_Alloc>::insert\(\)](#), [std::list<\\_Tp, \\_Alloc>::merge\(\)](#), [std::list<\\_Tp, \\_Alloc>::merge\(\)](#), [std::list<\\_Tp, \\_Alloc>::remove\(\)](#), [std::list<\\_Tp, \\_Alloc>::remove\\_if\(\)](#), [std::list<\\_Tp, \\_Alloc>::sort\(\)](#), [std::list<\\_Tp, \\_Alloc>::sort\(\)](#), [std::list<\\_Tp, \\_Alloc>::unique\(\)](#), and [std::list<\\_Tp, \\_Alloc>::unique\(\)](#).

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

```
template<typename _Tp, size_t _Nm>
_Tp * std::begin (
 _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the array.

**Parameters**

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Since

C++11



**begin()** [3/3]

```
template<typename _Container>
auto std::begin (
 const _Container & __cont) -> decltype(__cont.begin()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**boolalpha()**

```
ios_base & std::boolalpha (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::boolalpha)`.

References [std::ios\\_base::boolalpha](#).

**cbegin()**

```
template<typename _Container>
auto std::cbegin (
 const _Container & __cont) -> decltype(std::begin(__cont)) [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

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

Referenced by [std::vector<\\_Tp, \\_Alloc>::insert\(\)](#).

**cend()**

```
template<typename _Container>
auto std::cend (
 const _Container & __cont) -> decltype(std::end(__cont)) [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

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

**const\_pointer\_cast()**

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (
 const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
const_pointer_cast
```

**crbegin()**

```
template<typename _Container>
auto std::crbegin (
 const _Container & __cont) -> decltype(std::rbegin(__cont)) [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**Since**

C++14

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

**crend()**

```
template<typename _Container>
auto std::crend (
 const _Container & __cont) -> decltype(std::rend(__cont)) [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**Since**

C++14

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

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

```
template<typename _Container>
auto std::data (
 _Container & __cont) -> decltype(__cont.data()) [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of a container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++17

Referenced by [std::basic\\_stringbuf< char >::basic\\_stringbuf\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\(\)](#).

### data() [2/3]

```
template<typename _Tp, size_t _Nm>
_Tp * std::data (
 _Tp(&) __array[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an array.

#### Parameters

|                      |        |
|----------------------|--------|
| <code>__array</code> | Array. |
|----------------------|--------|

Since

C++17

### data() [3/3]

```
template<typename _Container>
auto std::data (
 const _Container & __cont) -> decltype(__cont.data()) [nodiscard], [constexpr],
[noexcept]
```

Return the data pointer of a const container.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++17

### dec()

```
ios_base & std::dec (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::dec, ios_base::basefield)`.  
References [std::ios\\_base::basefield](#), and [std::ios\\_base::dec](#).

### defaultfloat()

```
ios_base & std::defaultfloat (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::floatfield)`.  
References [std::ios\\_base::floatfield](#).

**distance()**

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

A generalization of pointer arithmetic.

**Parameters**

|                              |                                          |
|------------------------------|------------------------------------------|
| <code>__first, __last</code> | Input iterators that form a valid range. |
|------------------------------|------------------------------------------|

**Returns**

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`, or for random access iterators either `__last` is reachable from `__first` or `__first` is reachable from `__last`. In the latter case, `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and is constant time. For other iterator classes they are linear time.

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

Referenced by [\\_\\_merge\\_without\\_buffer\(\)](#), [\\_\\_sample\(\)](#), [std::deque< \\_Tp, \\_Alloc >::M\\_range\\_initialize\(\)](#), [is\\_heap\(\)](#), [is\\_heap\\_until\(\)](#), [is\\_heap\\_until\(\)](#), [std::sub\\_match< const char \\* >::length\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#), [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< node, leaf, head, inode, const\\_iterator, iterator, \\_Alloc >::num\\_children\(\)](#), [partition\\_point\(\)](#), [std::match\\_results< \\_BidirectionalIterator, polymorphic\\_allocator< sub\\_match< \\_BidirectionalIterator > > >::position\(\)](#), [\\_\\_gnu\\_cxx::random\\_sample\\_n\(\)](#), [\\_\\_gnu\\_cxx::random\\_sample\\_n\(\)](#), [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::size\(\)](#), and [std::list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::size\(\)](#).

**dynamic\_pointer\_cast()**

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (
 const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
```

`dynamic_pointer_cast`

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

```
template<typename _Container>
auto std::empty (
 const _Container & __cont) -> decltype(__cont.empty()) [nodiscard], [constexpr],
[noexcept]
```

Return whether a container is empty.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**Since**

C++17

Referenced by [std::list< \\_Tp, \\_Alloc >::sort\(\)](#), and [std::list< \\_Tp, \\_Alloc >::sort\(\)](#).

**empty()** [2/2]

```
template<typename _Tp, size_t _Nm>
bool std::empty (
 const _Tp(&)[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return whether an array is empty (always false).

**Parameters**

|   |           |
|---|-----------|
| - | An array. |
|---|-----------|

Since

C++17

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

```
template<typename _Container>
auto std::end (
 _Container & __cont) -> decltype(__cont.end()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the container.

**Parameters**

|        |            |
|--------|------------|
| __cont | Container. |
|--------|------------|

Since

C++11

Referenced by [cend\(\)](#), [std::vector<\\_Tp, \\_Alloc>::insert\(\)](#), [std::list<\\_Tp, \\_Alloc>::merge\(\)](#), [std::list<\\_Tp, \\_Alloc>::merge\(\)](#), [std::vector<\\_Tp, \\_Alloc>::operator=\(\)](#), [std::list<\\_Tp, \\_Alloc>::remove\(\)](#), [std::list<\\_Tp, \\_Alloc>::remove\\_if\(\)](#), [std::list<\\_Tp, \\_Alloc>::resize\(\)](#), [std::list<\\_Tp, \\_Alloc>::resize\(\)](#), [std::list<\\_Tp, \\_Alloc>::sort\(\)](#), [std::list<\\_Tp, \\_Alloc>::sort\(\)](#), [std::list<\\_Tp, \\_Alloc>::unique\(\)](#), and [std::list<\\_Tp, \\_Alloc>::unique\(\)](#).

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

```
template<typename _Tp, size_t _Nm>
_Tp * std::end (
 _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the array.

**Parameters**

|       |        |
|-------|--------|
| __arr | Array. |
|-------|--------|

Since

C++11

**end()** [3/3]

```
template<typename _Container>
auto std::end (
 const _Container & __cont) -> decltype(__cont.end()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++11

**endl()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::endl (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↔streambuf.buffering> for more on this subject.

References [flush\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::widen\(\)](#).

**ends()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::ends (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a null character into the output sequence.

*Null character* is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

References [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#).

**fabs()**

```
template<typename _Tp>
_Tp std::fabs (
 const std::complex< _Tp > & __z) [inline]
```

`fabs(__z)` TR1 8.1.8 [tr.c99.cmplx.fabs]

References [abs\(\)](#), and [fabs\(\)](#).

Referenced by [fabs\(\)](#).

**fixed()**

```
ios_base & std::fixed (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

References [std::ios\\_base::fixed](#), and [std::ios\\_base::floatfield](#).

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::flush (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

References [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#).

Referenced by [endl\(\)](#).

**from\_chars()**

```
template<typename _Tp, enable_if_t< __or< __is_signed_or_unsigned_integer< _Tp >, is_same<
char, _Tp > >::value, int > = 0>
from_chars_result std::from_chars (
 const char * __first,
 const char * __last,
 _Tp & __value,
 int __base = 10) [constexpr]
```

`std::from_chars` for integer types.

References [std::\\_\\_detail::\\_\\_from\\_chars\\_alnum\(\)](#), and [std::\\_\\_detail::\\_\\_from\\_chars\\_pow2\\_base\(\)](#).

**generate\_canonical()**

```
template<typename _RealT, size_t __digits, typename _Urbg>
_RealT std::generate_canonical (
 _Urbg & __urng)
```

Produce a random floating-point value in the range `[0..1)`

The result of `std::generate_canonical<RealT, digits>(urng)` is a random floating-point value of type `RealT` in the range `[0..1)`, using entropy provided by the uniform random bit generator `urng`. A value for `digits` may be passed to limit the precision of the result to so many bits, but normally `-1u` is passed to get the native precision of `RealT`. As many `urng()` calls are made as needed to obtain the required entropy. On rare occasions, more `urng()` calls are used. It is fastest when the value of `Urbg::max()` is a power of two less one, such as from a `std::philox4x32` (for float) or `philox4x64` (for double).

Since

C++11

References [std::\\_\\_numeric\\_limits\\_base::digits](#), [generate\\_canonical\(\)](#), and [std::\\_\\_numeric\\_limits\\_base::radix](#).

**get()** [1/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
 const pair< _Tp, _Up > && __p) [constexpr], [noexcept]
```

`std::get` overloads for accessing members of `std::pair`

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

**get()** [2/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
 const pair< _Tp, _Up > & __p) [constexpr], [noexcept]
```

`std::get` overloads for accessing members of `std::pair`

**get()** [3/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
 const pair< _Tp1, _Tp2 > && __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

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

**get()** [4/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
 const pair< _Tp1, _Tp2 > & __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [5/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
 const pair< _Up, _Tp > && __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

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

**get()** [6/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
 const pair< _Up, _Tp > & __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [7/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
 pair< _Tp, _Up > && __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

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

**get()** [8/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
 pair< _Tp, _Up > & __p) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

**get()** [9/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
 pair< _Tp1, _Tp2 > && __in) [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

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



**get()** [10/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
 pair< _Tp1, _Tp2 > & __in) [constexpr], [noexcept]
std::get overloads for accessing members of std::pair
```

**get()** [11/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
 pair< _Up, _Tp > && __p) [constexpr], [noexcept]
std::get overloads for accessing members of std::pair
References forward\(\).
```

**get()** [12/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
 pair< _Up, _Tp > & __p) [constexpr], [noexcept]
std::get overloads for accessing members of std::pair
```

**get\_money()**

```
template<typename _MoneyT>
_Get_money< _MoneyT > std::get_money (
 _MoneyT & __mon,
 bool __intl = false) [inline]
Extended manipulator for extracting money.
```

**Parameters**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__mon</code>  | Either long double or a specialization of <code>basic_string</code> . |
| <code>__intl</code> | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator extracts `__mon`.

**get\_new\_handler()**

```
new_handler std::get_new_handler () [noexcept]
Return the current new handler.
```

**get\_temporary\_buffer()**

```
template<typename _Tp>
pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (
 ptrdiff_t __len) [noexcept]
Allocates a temporary buffer.
```

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__len</code> | The number of objects of type <code>Tp</code> . |
|--------------------|-------------------------------------------------|

**Returns**

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A pair<> is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`), or a pair of 0 values if no storage can be obtained*. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

**get\_time()**

```
template<typename _CharT>
_Get_time< _CharT > std::get_time (
 std::tm * __tmb,
 const _CharT * __fmt) [inline]
```

Extended manipulator for extracting time.

This manipulator uses `time_get::get` to extract time. [ext.manip]

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__tmb</code> | struct to extract the time data to. |
| <code>__fmt</code> | format string.                      |

**getline()** [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > && __is,
 basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Read a line from an rvalue stream into a string.

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

**getline()** [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > && __is,
 basic_string< _CharT, _Traits, _Alloc > & __str,
 _CharT __delim) [inline]
```

Read a line from an rvalue stream into a string.

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

**getline()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Read a line from stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

References [getline\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::widen\(\)](#).

**getline()** [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 _CharT __delim)
Read a line from stream into a string.
```

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

References [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::append\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>::max\\_size\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**getline()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
Read a line from stream into a string.
```

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased.

If end of line is encountered, it is extracted but not stored into `__str`.

References [getline\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits >::widen\(\)](#).

**getline()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str,
 _CharT __delim)
```

Read a line from stream into a string.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc >::erase\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc >::max\\_size\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits >::setstate\(\)](#).

Referenced by [getline\(\)](#), [getline\(\)](#), [getline\(\)](#), and [getline\(\)](#).

**hex()**

```
ios_base & std::hex (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

References [std::ios\\_base::basefield](#), and [std::ios\\_base::hex](#).

**hexfloat()**

```
ios_base & std::hexfloat (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed|ios_base::scientific, ios_base::floatfield)`.

References [std::ios\\_base::fixed](#), [std::ios\\_base::floatfield](#), and [std::ios\\_base::scientific](#).

**inout\_ptr()**

```
template<typename _Pointer = void, typename _Smart, typename... _Args>
auto std::inout_ptr (
 _Smart & __s,
 _Args &&... __args) [inline]
```

Adapt a smart pointer for functions taking an inout pointer parameter.

**Template Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>_Pointer</code> | The type of pointer to convert to. |
|-----------------------|------------------------------------|

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The pointer that should take ownership of the result.      |
| <code>__args</code> | Pack of arguments to use when resetting the smart pointer. |

**Returns**

A `std::inout_ptr_t` referring to `__s`.

**Since**

C++23

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

**internal()**

```
ios_base & std::internal (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.  
References [std::ios\\_base::adjustfield](#), and [std::ios\\_base::internal](#).

**isalnum()**

```
template<typename _CharT>
bool std::isalnum (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.  
References [use\\_facet\(\)](#).

**isalpha()**

```
template<typename _CharT>
bool std::isalpha (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.  
References [use\\_facet\(\)](#).

**isblank()**

```
template<typename _CharT>
bool std::isblank (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::blank, __c)`.

References [use\\_facet\(\)](#).

**iscntrl()**

```
template<typename _CharT>
bool std::iscntrl (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

References [use\\_facet\(\)](#).

**isdigit()**

```
template<typename _CharT>
bool std::isdigit (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

References [use\\_facet\(\)](#).

**isgraph()**

```
template<typename _CharT>
bool std::isgraph (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

References [use\\_facet\(\)](#).

**islower()**

```
template<typename _CharT>
bool std::islower (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

References [use\\_facet\(\)](#).

**isprint()**

```
template<typename _CharT>
bool std::isprint (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::print, __c)`.

References [use\\_facet\(\)](#).

**ispunct()**

```
template<typename _CharT>
bool std::ispunct (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `cctype.is(cctype_base::punct, __c)`.

References [use\\_facet\(\)](#).

**isspace()**

```
template<typename _CharT>
bool std::isspace (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `cctype.is(cctype_base::space, __c)`.

References [use\\_facet\(\)](#).

**isupper()**

```
template<typename _CharT>
bool std::isupper (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `cctype.is(cctype_base::upper, __c)`.

References [use\\_facet\(\)](#).

**isxdigit()**

```
template<typename _CharT>
bool std::isxdigit (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `cctype.is(cctype_base::xdigit, __c)`.

References [use\\_facet\(\)](#).

**left()**

```
ios_base & std::left (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

References [std::ios\\_base::adjustfield](#), and [std::ios\\_base::left](#).

**noboolalpha()**

```
ios_base & std::noboolalpha (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::boolalpha)`.

References [std::ios\\_base::boolalpha](#).

**noshowbase()**

```
ios_base & std::noshowbase (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showbase)`.

References [std::ios\\_base::showbase](#).

**noshowpoint()**

```
ios_base & std::noshowpoint (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showpoint)`.

References [std::ios\\_base::showpoint](#).

**noshowpos()**

```
ios_base & std::noshowpos (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::showpos)`.

References [std::ios\\_base::showpos](#).

**noskipws()**

```
ios_base & std::noskipws (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::skipws)`.

References [std::ios\\_base::skipws](#).

**nounitbuf()**

```
ios_base & std::nounitbuf (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::unitbuf)`.

References [std::ios\\_base::unitbuf](#).

**nouppercase()**

```
ios_base & std::nouppercase (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::uppercase)`.

References [std::ios\\_base::uppercase](#).

**oct()**

```
ios_base & std::oct (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

References [std::ios\\_base::basefield](#), and [std::ios\\_base::oct](#).

**operator"!=()" [1/5]**

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator!= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on `operator==`.

**operator"!=()" [2/5]**

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```



Returns `!(x == y)`.

#### **operator!=()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on `operator==`.

#### **operator!=()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns `!(x == y)`.

#### **operator!=()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on `operator==`.

#### **operator&()**

```
template<size_t _Nb>
bitset< _Nb > std::operator& (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

#### **Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A bitset.                                       |
| <code>__y</code> | A bitset of the same size as <code>__x</code> . |

#### **Returns**

A new bitset.

These should be self-explanatory.

#### **operator+()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
 _CharT __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate character and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator+()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const _CharT * __lhs,
 const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate C string and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator+()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const basic_string<_CharT, _Traits, _Alloc> & __lhs,
 _CharT __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate string and character.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::get\\_allocator\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator+()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate string and C string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::get\\_allocator\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::size\(\)](#).

**operator+()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr]
```

Concatenate two strings.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::get\\_allocator\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::size\(\)](#).

**operator<()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator< (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap ordering relation.

**Parameters**

|                  |             |
|------------------|-------------|
| <code>__x</code> | A multimap. |
|------------------|-------------|

|                                    |                                                   |
|------------------------------------|---------------------------------------------------|
| <code>_↔</code><br><code>_y</code> | A multimap of the same type as <code>__x</code> . |
|------------------------------------|---------------------------------------------------|

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with `<`. See `std::lexicographical_compare()` for how the determination is made.

**operator<()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset ordering relation.

**Parameters**

|                                    |                                                   |
|------------------------------------|---------------------------------------------------|
| <code>_↔</code><br><code>_x</code> | A multiset.                                       |
| <code>_↔</code><br><code>_y</code> | A multiset of the same type as <code>__x</code> . |

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with `<`. See `std::lexicographical_compare()` for how the determination is made.

**operator<()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Queue ordering relation.

**Parameters**

|                                    |                                        |
|------------------------------------|----------------------------------------|
| <code>_↔</code><br><code>_x</code> | A queue.                               |
| <code>_↔</code><br><code>_y</code> | A queue of the same type as <i>x</i> . |

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

References [std::queue<\\_Tp, \\_Sequence>::c](#).

**operator<()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set ordering relation.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | A set.                       |
| <code>__y</code> | A set of the same type as x. |

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with `<`. See `std::lexicographical_compare()` for how the determination is made.

**operator<()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Stack ordering relation.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | A stack.                       |
| <code>__y</code> | A stack of the same type as x. |

**Returns**

True iff `x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

**operator<<()** [1/14]

```
template<typename _Ostream, typename _Tp>
__rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (
 _Ostream && __os,
 const _Tp & __x) [inline]
```

Generic inserter for rvalue stream.

**Parameters**

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__os</code> | An input stream.                          |
| <code>__x</code>  | A reference to the object being inserted. |

**Returns**

`__os`

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

**operator<<()** [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Write string to a stream.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__os</code>  | Output stream.       |
| <code>__str</code> | String to write out. |

**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

**operator<<()** [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Write string to a stream.

**Parameters**

|                   |                |
|-------------------|----------------|
| <code>__os</code> | Output stream. |
|-------------------|----------------|

|                    |                      |
|--------------------|----------------------|
| <code>__str</code> | String to write out. |
|--------------------|----------------------|

**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

**operator<<() [4/14]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 _CharT __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<() [5/14]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [6/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 const _CharT * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [7/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 const char * __s)
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2]). `__out.width(0)` is then called.



**operator<<()** [8/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
 basic_ostream< char, _Traits > & __out,
 char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

**operator<<()** [9/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
 basic_ostream< char, _Traits > & __out,
 const char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

**operator<<()** [10/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
```

```
basic_ostream< char, _Traits > & __out,
const signed char * __s) [inline]
```

String inserters.

#### Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

#### Returns

`out`

#### Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

#### **operator<<()** [11/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
 basic_ostream< char, _Traits > & __out,
 const unsigned char * __s) [inline]
```

String inserters.

#### Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

#### Returns

`out`

#### Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

#### **operator<<()** [12/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
 basic_ostream< char, _Traits > & __out,
 signed char __c) [inline]
```

Character inserters.

#### Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

#### Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

#### `operator<<()` [13/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
 basic_ostream< char, _Traits > & __out,
 unsigned char __c) [inline]
```

Character inserters.

#### Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

#### Returns

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

#### `operator<<()` [14/14]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

References `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_ios< _CharT, _Traits`

Referenced by `std::shared_ptr< _State_base >::operator<<()`.

**operator<=()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator<= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

**operator<=()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(y < x).

**operator<=()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator<=()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(y < x).

**operator<=()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator<=>()** [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
[nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__lhs</code> | A string.                 |
| <code>__rhs</code> | A null-terminated string. |

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [2/7]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) -> decltype(__detail::__↵
char_traits_cmp_cat< _Traits >(0)) [nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| <code>__lhs</code> | A string.                 |
| <code>__rhs</code> | A null-terminated string. |

**Returns**

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

**operator<=>() [3/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Deque ordering relation.

**Parameters**

|                          |                                                |
|--------------------------|------------------------------------------------|
| <code>__↵<br/>__x</code> | A deque.                                       |
| <code>__↵<br/>__y</code> | A deque of the same type as <code>__x</code> . |

**Returns**

A value indicating whether `__x` is less than, equal to, greater than, or incomparable with `__y`.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

**operator<=>() [4/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
 const forward_list< _Tp, _Alloc > & __x,
 const forward_list< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Forward list ordering relation.

**Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $\_x$ | A forward_list.                            |
| $\_y$ | A forward_list of the same type as $\_x$ . |

**Returns**

A value indicating whether  $\_x$  is less than, equal to, greater than, or incomparable with  $\_y$ .

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

**operator<=>() [5/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

List ordering relation.

**Parameters**

|       |                                    |
|-------|------------------------------------|
| $\_x$ | A list.                            |
| $\_y$ | A list of the same type as $\_x$ . |

**Returns**

A value indicating whether  $\_x$  is less than, equal to, greater than, or incomparable with  $\_y$ .

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::end()`, and `std::list< _Tp, _Alloc >::size()`.

**operator<=>() [6/7]**

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
__detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map ordering relation.

**Parameters**

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A map.                            |
| $\_y$ | A map of the same type as $\_x$ . |

**Returns**

A value indicating whether `__x` is less than, equal to, greater than, or incomparable with `__y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`. See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

**operator<=>() [7/7]**

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [nodiscard], [constexpr]
```

Vector ordering relation.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A vector.                                       |
| <code>__y</code> | A vector of the same type as <code>__x</code> . |

**Returns**

A value indicating whether `__x` is less than, equal to, greater than, or incomparable with `__y`.

See `std::lexicographical_compare_three_way()` for how the determination is made. This operator is used to synthesize relational operators like `<` and `>=` etc.

References [std::vector<\\_Tp, \\_Alloc>::begin\(\)](#), [std::vector<\\_Tp, \\_Alloc>::end\(\)](#), and [lexicographical\\_compare\\_three\\_way\(\)](#).

**operator==() [1/14]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator==(
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline], [nodiscard], [constexpr]
```

Test equivalence of string and C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**operator==() [2/14]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator==(
```

```
const basic_string< _CharT, _Traits, _Alloc > & __lhs,
const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [nodiscard], [constexpr],
[noexcept]
```

Test equivalence of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

#### `operator==()` [3/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline], [nodiscard]
```

Deque equality comparison.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A deque.                                       |
| <code>__y</code> | A deque of the same type as <code>__x</code> . |

#### Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

#### `operator==()` [4/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly)
```

Forward list equality comparison.

#### Parameters

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__lx</code> | A <code>forward_list</code>                                         |
| <code>__ly</code> | A <code>forward_list</code> of the same type as <code>__lx</code> . |



**Returns**

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

References [std::forward\\_list<\\_Tp, \\_Alloc >::cbegin\(\)](#), and [std::forward\\_list<\\_Tp, \\_Alloc >::cend\(\)](#).

**operator==(** [5/14]

```
template<typename _StateT>
bool std::operator==(
 const fpos<_StateT> & __lhs,
 const fpos<_StateT> & __rhs) [inline]
```

Test if equivalent to another position.

**operator==(** [6/14]

```
template<typename _Res, typename... _Args>
bool std::operator==(
 const function<_Res(_Args...)> & __f,
 nullptr_t) [inline], [noexcept]
```

Test whether a polymorphic function object wrapper is empty.

**Returns**

true if the wrapper has no target, false otherwise

This function will not throw exceptions.

**operator==(** [7/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator==(
 const list<_Tp, _Alloc> & __x,
 const list<_Tp, _Alloc> & __y) [inline], [nodiscard]
```

List equality comparison.

**Parameters**

|       |                                    |
|-------|------------------------------------|
| $\_x$ | A list.                            |
| $\_y$ | A list of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==(** [8/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator==(
```

```
const map< _Key, _Tp, _Compare, _Alloc > & __x,
const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map equality comparison.

#### Parameters

|       |                              |
|-------|------------------------------|
| $\_x$ | A map.                       |
| $\_y$ | A map of the same type as x. |

#### Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

#### **operator==()** [9/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap equality comparison.

#### Parameters

|       |                                     |
|-------|-------------------------------------|
| $\_x$ | A multimap.                         |
| $\_y$ | A multimap of the same type as __x. |

#### Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

#### **operator==()** [10/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset equality comparison.

#### Parameters

|       |             |
|-------|-------------|
| $\_x$ | A multiset. |
|-------|-------------|

|                           |                                        |
|---------------------------|----------------------------------------|
| $\_x \leftrightarrow \_y$ | A multiset of the same type as $\_x$ . |
|---------------------------|----------------------------------------|

**Returns**

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==( ) [11/14]**

```
template<typename _Tp, typename _Seq>
bool std::operator== (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Queue equality comparison.

**Parameters**

|                           |                                     |
|---------------------------|-------------------------------------|
| $\_x \leftrightarrow \_x$ | A queue.                            |
| $\_x \leftrightarrow \_y$ | A queue of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

References [std::queue< \\_Tp, \\_Sequence >::c](#).

**operator==( ) [12/14]**

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set equality comparison.

**Parameters**

|                           |                                   |
|---------------------------|-----------------------------------|
| $\_x \leftrightarrow \_x$ | A set.                            |
| $\_x \leftrightarrow \_y$ | A set of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

**operator==()** [13/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Stack equality comparison.

**Parameters**

|       |                                     |
|-------|-------------------------------------|
| $\_x$ | A stack.                            |
| $\_y$ | A stack of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

**operator==()** [14/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline], [nodiscard], [constexpr]
```

Vector equality comparison.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_x$ | A vector.                            |
| $\_y$ | A vector of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

References `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, and `std::vector< _Tp, _Alloc >::size()`.

**operator>()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator> (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

**operator>()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $y < x$ .

**operator>()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $y < x$ .

**operator>()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>=()** [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator>= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

**operator>=()** [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $!(x < y)$ .

**operator>=()** [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>=()** [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x < y).

**operator>=()** [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline], [nodiscard]
```

Based on operator<.

**operator>>()** [1/10]

```
template<typename _Istream, typename _Tp>
__rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (
 _Istream && __is,
 _Tp && __x) [inline]
```

Generic extractor for rvalue stream.

**Parameters**

|                   |                                       |
|-------------------|---------------------------------------|
| <code>__is</code> | An input stream.                      |
| <code>__x</code>  | A reference to the extraction target. |

**Returns**

`__is`

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

References [forward\(\)](#), and [move\(\)](#).

**operator>>()** [2/10]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __in,
 _CharT & __c)
```

Character extractors.

**Parameters**

|                          |                        |
|--------------------------|------------------------|
| $\leftrightarrow$<br>_in | An input stream.       |
| $\leftrightarrow$<br>_c  | A character reference. |

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

References [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rd](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**operator>>()** [3/10]

```
template<typename _CharT, typename _Traits, size_t _Num>
basic_istream<_CharT, _Traits> & std::operator>> (
 basic_istream<_CharT, _Traits> & __in,
 _CharT(&) __s[_Num]) [inline]
```

Character string extractors.

**Parameters**

|                          |                                                            |
|--------------------------|------------------------------------------------------------|
| $\leftrightarrow$<br>_in | An input stream.                                           |
| $\leftrightarrow$<br>_s  | A character array (or a pointer to an array before C++20). |

**Returns**

\_\_in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**operator>>()** [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)
Read stream into a string.
```

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

References `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `use_facet()`, and `std::ios_base::width()`.

**operator>>()** [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str)
Read stream into a string.
```

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::ios_base::getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `use_facet()`, and `std::ios_base::width()`.

**operator>>()** [6/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
```



```
basic_istream< char, _Traits > & __in,
signed char & __c) [inline]
```

Character extractors.

#### Parameters

|                      |                        |
|----------------------|------------------------|
| $\leftarrow$<br>__in | An input stream.       |
| $\leftarrow$<br>__c  | A character reference. |

#### Returns

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

#### `operator>>()` [7/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
 basic_istream< char, _Traits > & __in,
 signed char(&) __s[_Num]) [inline]
```

Character string extractors.

#### Parameters

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| $\leftarrow$<br>__in | An input stream.                                           |
| $\leftarrow$<br>__s  | A character array (or a pointer to an array before C++20). |

#### Returns

\_\_in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

**operator>>()** [8/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
 basic_istream< char, _Traits > & __in,
 unsigned char & __c) [inline]
```

Character extractors.

**Parameters**

|                                                                                           |                        |
|-------------------------------------------------------------------------------------------|------------------------|
| <br>__in | An input stream.       |
| <br>__c  | A character reference. |

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

**operator>>()** [9/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
 basic_istream< char, _Traits > & __in,
 unsigned char(&) __s[_Num]) [inline]
```

Character string extractors.

**Parameters**

|                                                                                             |                                                            |
|---------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <br>__in | An input stream.                                           |
| <br>__s  | A character array (or a pointer to an array before C++20). |

**Returns**

\_\_in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored

- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

### **operator>>()** [10/10]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_istream< _CharT, _Traits > & std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

### **operator^()**

```
template<size_t _Nb>
bitset< _Nb > std::operator^ (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

#### **Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| $\leftarrow$<br>__x | A bitset.                         |
| $\leftarrow$<br>__y | A bitset of the same size as __x. |

#### **Returns**

A new bitset.

These should be self-explanatory.

### **operator" | ()**

```
template<size_t _Nb>
bitset< _Nb > std::operator| (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

#### **Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| $\leftarrow$<br>__x | A bitset.                         |
| $\leftarrow$<br>__y | A bitset of the same size as __x. |

**Returns**

A new bitset.

These should be self-explanatory.

**out\_ptr()**

```
template<typename _Pointer = void, typename _Smart, typename... _Args>
auto std::out_ptr (
 _Smart & __s,
 _Args &&... __args) [inline]
```

Adapt a smart pointer for functions taking an output pointer parameter.

**Template Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>_Pointer</code> | The type of pointer to convert to. |
|-----------------------|------------------------------------|

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The pointer that should take ownership of the result.      |
| <code>__args</code> | Pack of arguments to use when resetting the smart pointer. |

**Returns**

A `std::out_ptr_t` referring to `__s`.

**Since**

C++23

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

**put\_money()**

```
template<typename _MoneyT>
_Put_money< _MoneyT > std::put_money (
 const _MoneyT & __mon,
 bool __intl = false) [inline]
```

Extended manipulator for inserting money.

**Parameters**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__mon</code>  | Either long double or a specialization of <code>basic_string</code> . |
| <code>__intl</code> | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator inserts `__mon`.

**put\_time()**

```
template<typename _CharT>
_Put_time< _CharT > std::put_time (
```

```
const std::tm * __tmb,
const _CharT * __fmt) [inline]
```

Extended manipulator for formatting time.

This manipulator uses `time_put::put` to format time. [ext.manip]

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__tmb</code> | struct tm time data to format. |
| <code>__fmt</code> | format string.                 |

#### **rbegin()** [1/4]

```
template<typename _Container>
auto std::rbegin (
 _Container & __cont) -> decltype(__cont.rbegin()) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the container.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

Referenced by [crbegin\(\)](#).

#### **rbegin()** [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rbegin (
 _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the array.

#### Parameters

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Since

C++14

#### **rbegin()** [3/4]

```
template<typename _Container>
auto std::rbegin (
 const _Container & __cont) -> decltype(__cont.rbegin()) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

**rbegin()** [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rbegin (
 initializer_list< _Tp > __il) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the `initializer_list`.**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__il</code> | <code>initializer_list</code> . |
|-------------------|---------------------------------|

Since

C++14

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

```
template<typename _Container>
auto std::rend (
 _Container & __cont) -> decltype(__cont.rend()) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

Referenced by [crend\(\)](#).**rend()** [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rend (
 _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the array.

**Parameters**

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Since

C++14

### rend() [3/4]

```
template<typename _Container>
auto std::rend (
 const _Container & __cont) -> decltype(__cont.rend()) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++14

### rend() [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rend (
 initializer_list< _Tp > __il) [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the initializer\_list.

#### Parameters

|                   |                   |
|-------------------|-------------------|
| <code>__il</code> | initializer_list. |
|-------------------|-------------------|

Since

C++14

### replace\_copy()

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

Copy a sequence, replacing each element of one value with another value.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

|                          |                           |
|--------------------------|---------------------------|
| <code>__result</code>    | An output iterator.       |
| <code>__old_value</code> | The value to be replaced. |
| <code>__new_value</code> | The replacement value.    |

### Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[__first,__last)` to the output range `[__result,__result+(__last-__first))` replacing elements equal to `__old_value` with `__new_value`.

### resetiosflags()

```
_Resetiosflags std::resetiosflags (
 ios_base::fmtflags __mask) [inline]
```

Manipulator for `setf`.

### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0,__mask)`.

### return\_temporary\_buffer()

```
template<typename _Tp>
void std::return_temporary_buffer (
 _Tp * __p) [inline]
```

The companion to `get_temporary_buffer()`.

### Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__p</code> | A buffer previously allocated by <code>get_temporary_buffer</code> . |
|------------------|----------------------------------------------------------------------|

Frees the memory pointed to by `__p`.

### right()

```
ios_base & std::right (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

References `std::ios_base::adjustfield`, and `std::ios_base::right`.

### sample()

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _URBG
UniformRandomBitGenerator>
_SampleIterator std::sample (
 _PopulationIterator __first,
 _PopulationIterator __last,
 _SampleIterator __out,
```



```

 _Distance __n,
 _UniformRandomBitGenerator && __g)

```

Take a random sample from a population.

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

### **satürating\_add()**

```

template<typename _Tp>
_Tp std::satürating_add (
 _Tp __x,
 _Tp __y) [constexpr], [noexcept]

```

Add two integers, with saturation in case of overflow.

### **satürating\_cast()**

```

template<typename _Res, typename _Tp>
_Res std::satürating_cast (
 _Tp __x) [constexpr], [noexcept]

```

Divide one integer by another, with saturation in case of overflow.

### **satürating\_div()**

```

template<typename _Tp>
_Tp std::satürating_div (
 _Tp __x,
 _Tp __y) [constexpr], [noexcept]

```

Divide one integer by another, with saturation in case of overflow.

### **satürating\_mul()**

```

template<typename _Tp>
_Tp std::satürating_mul (
 _Tp __x,
 _Tp __y) [constexpr], [noexcept]

```

Multiply two integers, with saturation in case of overflow.

### **satürating\_sub()**

```

template<typename _Tp>
_Tp std::satürating_sub (
 _Tp __x,
 _Tp __y) [constexpr], [noexcept]

```

Subtract one integer from another, with saturation in case of overflow.

### **scientific()**

```

ios_base & std::scientific (
 ios_base & __base) [inline]

```

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

References [std::ios\\_base::floatfield](#), and [std::ios\\_base::scientific](#).

### **search()**

```

template<typename _ForwardIterator, typename _Searcher>
_FwdIterator std::search (

```

```

 _ForwardIterator __first,
 _ForwardIterator __last,
 const _Searcher & __searcher) [inline], [nodiscard], [constexpr]

```

Search a sequence using a Searcher object.

#### Parameters

|                         |                     |
|-------------------------|---------------------|
| <code>__first</code>    | A forward iterator. |
| <code>__last</code>     | A forward iterator. |
| <code>__searcher</code> | A callable object.  |

#### Returns

```
__searcher(__first, __last).first
```

#### set\_new\_handler()

```

new_handler std::set_new_handler (
 new_handler) throw ()

```

Takes a replacement handler as the argument, returns the previous handler.

#### setbase()

```

_Setbase std::setbase (
 int __base) [inline]

```

Manipulator for `setf`.

#### Parameters

|                     |                 |
|---------------------|-----------------|
| <code>__base</code> | A numeric base. |
|---------------------|-----------------|

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when `base` is 8, 10, or 16, accordingly, and to 0 if `__base` is any other value.

#### setfill()

```

template<typename _CharT>
_Setfill< _CharT > std::setfill (
 _CharT __c) [inline]

```

Manipulator for `fill`.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The new fill character. |
|------------------|-------------------------|

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

#### setiosflags()

```

_Setiosflags std::setiosflags (
 ios_base::fmtflags __mask) [inline]

```

Manipulator for `setf`.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator sets the format flags to `__mask`.

#### setprecision()

```
_Setprecision std::setprecision (
 int __n) [inline]
```

Manipulator for `precision`.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__n</code> | The new precision. |
|------------------|--------------------|

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

#### setw()

```
_Setw std::setw (
 int __n) [inline]
```

Manipulator for `width`.

#### Parameters

|                  |                |
|------------------|----------------|
| <code>__n</code> | The new width. |
|------------------|----------------|

Sent to a stream object, this manipulator calls `width(__n)` for that object.

#### showbase()

```
ios_base & std::showbase (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showbase)`.

References [std::ios\\_base::showbase](#).

#### showpoint()

```
ios_base & std::showpoint (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpoint)`.

References [std::ios\\_base::showpoint](#).

#### showpos()

```
ios_base & std::showpos (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpos)`.

References [std::ios\\_base::showpos](#).

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

```
template<typename _Container>
auto std::size (
 const _Container & __cont) -> decltype(__cont.size()) [nodiscard], [constexpr],
[noexcept]
```

Return the size of a container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

**Since**

C++17

Referenced by [std::basic\\_stringbuf< char >::basic\\_stringbuf\(\)](#), [std::deque< \\_Tp, \\_Alloc >::\\_M\\_new\\_elements\\_at\\_back\(\)](#), [std::deque< \\_Tp, \\_Alloc >::\\_M\\_new\\_elements\\_at\\_front\(\)](#), [std::deque< \\_Tp, \\_Alloc >::\\_M\\_push\\_back\\_aux\(\)](#), [std::deque< \\_Tp, \\_Alloc >::find\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\\_first\\_of\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\\_last\\_not\\_of\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::find\\_last\\_of\(\)](#), [std::deque< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::vector< \\_Tp, \\_Alloc >::operator=\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::reserve\(\)](#), [std::vector< \\_Tp, \\_Alloc >::reserve\(\)](#), [std::vector< \\_State< \\_TraitsT::char\\_type >, \\_Alloc >::reserve\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::rfind\(\)](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::rfind\(\)](#).

**size()** [2/2]

```
template<typename _Tp, size_t _Nm>
size_t std::size (
 const _Tp(&)[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return the size of an array.

**Parameters**

|   |           |
|---|-----------|
| - | An array. |
|---|-----------|

**Returns**

The number of elements in the array.

**Since**

C++17

**skipws()**

```
ios_base & std::skipws (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::skipws)`.

References [std::ios\\_base::skipws](#).

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

```
template<typename _Container>
auto std::ssize (
 const _Container & __cont) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__cont.size())>> > [nodiscard], [constexpr], [noexcept]
```

Return the size of a container, as a signed integer.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Since

C++20

**ssize()** [2/2]

```
template<typename _Tp, ptrdiff_t _Num>
ptrdiff_t std::ssize (
 const _Tp(&)[_Num]) [nodiscard], [constexpr], [noexcept]
```

Return the length of an array, as a signed integer.

Since

C++20

**static\_pointer\_cast()**

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::static_pointer_cast (
 const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
static_pointer_cast
```

**swap()** [1/19]

```
template<class _CharT, class _Traits>
void std::swap (
 basic_filebuf< _CharT, _Traits > & __x,
 basic_filebuf< _CharT, _Traits > & __y) [inline]
```

Swap specialization for filebufs.

**swap()** [2/19]

```
template<class _CharT, class _Traits>
void std::swap (
 basic_fstream< _CharT, _Traits > & __x,
 basic_fstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for fstreams.

**swap()** [3/19]

```
template<class _CharT, class _Traits>
void std::swap (
```

```

 basic_ifstream< _CharT, _Traits > & __x,
 basic_ifstream< _CharT, _Traits > & __y) [inline]

```

Swap specialization for ifstreams.

#### swap() [4/19]

```

template<class _CharT, class _Traits, class _Allocator>
void std::swap (
 basic_istream< _CharT, _Traits, _Allocator > & __x,
 basic_istream< _CharT, _Traits, _Allocator > & __y) [inline]

```

Swap specialization for istreams.

#### swap() [5/19]

```

template<class _CharT, class _Traits>
void std::swap (
 basic_ofstream< _CharT, _Traits > & __x,
 basic_ofstream< _CharT, _Traits > & __y) [inline]

```

Swap specialization for ofstreams.

#### swap() [6/19]

```

template<class _CharT, class _Traits, class _Allocator>
void std::swap (
 basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
 basic_ostringstream< _CharT, _Traits, _Allocator > & __y) [inline]

```

Swap specialization for ostringstreams.

#### swap() [7/19]

```

template<typename _CharT, typename _Traits, typename _Alloc>
void std::swap (
 basic_string< _CharT, _Traits, _Alloc > & __lhs,
 basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [constexpr], [noexcept]

```

Swap contents of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

#### swap() [8/19]

```

template<class _CharT, class _Traits, class _Allocator>
void std::swap (
 basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
 basic_stringbuf< _CharT, _Traits, _Allocator > & __y) [inline], [noexcept]

```

Swap specialization for stringbufs.

#### swap() [9/19]

```

template<class _CharT, class _Traits, class _Allocator>
void std::swap (

```

```
basic_stringstream< _CharT, _Traits, _Allocator > & __x,
basic_stringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for stringstream.

#### swap() [10/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
 deque< _Tp, _Alloc > & __x,
 deque< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::deque::swap()`.

#### swap() [11/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
 forward_list< _Tp, _Alloc > & __lx,
 forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See `std::forward_list::swap()`.

#### swap() [12/19]

```
template<typename _Res, typename... _Args>
void std::swap (
 function< _Res(_Args...) > & __x,
 function< _Res(_Args...) > & __y) [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.

This function will not throw exceptions.

#### swap() [13/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
 list< _Tp, _Alloc > & __x,
 list< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::list::swap()`.

#### swap() [14/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
 map< _Key, _Tp, _Compare, _Alloc > & __x,
 map< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::map::swap()`.

References [std::pair< \\_T1, \\_T2 >::swap\(\)](#).

#### swap() [15/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
 multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::multimap::swap()`.

References [std::multimap< \\_Key, \\_Tp, \\_Compare, \\_Alloc >::swap\(\)](#).

**swap()** [16/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
 multiset< _Key, _Compare, _Alloc > & __x,
 multiset< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::multiset::swap()`.

References [std::multiset< \\_Key, \\_Compare, \\_Alloc >::swap\(\)](#).

**swap()** [17/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
 set< _Key, _Compare, _Alloc > & __x,
 set< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::set::swap()`.

References [std::set< \\_Key, \\_Compare, \\_Alloc >::swap\(\)](#).

**swap()** [18/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
 vector< _Tp, _Alloc > & __x,
 vector< _Tp, _Alloc > & __y) [inline], [constexpr], [noexcept]
```

See `std::vector::swap()`.

**tolower()**

```
template<typename _CharT>
_CharT std::tolower (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype::tolower(__c)`.

References [tolower\(\)](#), and [use\\_facet\(\)](#).

Referenced by [tolower\(\)](#).

**toupper()**

```
template<typename _CharT>
_CharT std::toupper (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype::toupper(__c)`.

References [toupper\(\)](#), and [use\\_facet\(\)](#).

Referenced by [toupper\(\)](#).

**unitbuf()**

```
ios_base & std::unitbuf (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::unitbuf)`.

References [std::ios\\_base::unitbuf](#).



**uppercase()**

```
ios_base & std::uppercase (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::uppercase)`.

References [std::ios\\_base::uppercase](#).

**ws()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::ws (
 basic_istream< _CharT, _Traits > & __is)
```

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator<>> on cin and your object. Note that the same effect can be achieved by creating a `std::basic_istream::sentry` inside your definition of operator<>>.

References [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [use\\_facet\(\)](#).

**4.10.5 Variable Documentation****cerr**

```
ostream std::cerr [extern]
```

Linked to standard error (unbuffered).

**cin**

```
istream std::cin [extern]
```

Linked to standard input.

**clog**

```
ostream std::clog [extern]
```

Linked to standard error (buffered).

**cout**

```
ostream std::cout [extern]
```

Linked to standard output.

**default\_sentinel**

```
default_sentinel_t std::default_sentinel [inline], [constexpr]
```

A default sentinel value.

**destroying\_delete**

```
destroying_delete_t std::destroying_delete [inline], [constexpr]
```

Tag variable of type `destroying_delete_t`.

## ignore

`_Swallow_assign std::ignore [inline], [constexpr]`

Used with `std::tie` to ignore an element of a tuple

When using `std::tie` to assign the elements of a tuple to variables, unwanted elements can be ignored by using `std::ignore`. For example:

```
int x, y;
std::tie(x, std::ignore, y) = std::make_tuple(1, 2, 3);
```

This assignment will perform `x=1; std::ignore=2; y=3;` which results in the second element being ignored.

Since

C++11

## wcerr

`wostream std::wcerr [extern]`

Linked to standard error (unbuffered).

## wcin

`wistream std::wcin [extern]`

Linked to standard input.

## wclog

`wostream std::wclog [extern]`

Linked to standard error (buffered).

## wcout

`wostream std::wcout [extern]`

Linked to standard output.

## 4.11 std::\_\_debug Namespace Reference

### Classes

- class [bitset](#)
- class [deque](#)
- class [forward\\_list](#)
- class [list](#)
- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- class [vector](#)

## Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**deque** (`_InputIterator, _InputIterator, _Allocator=_Allocator()`) -> `deque<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
**deque** (`size_t, _Tp, _Allocator=_Allocator()`) -> `deque<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**forward\_list** (`_InputIterator, _InputIterator, _Allocator=_Allocator()`) -> `forward_list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
**forward\_list** (`size_t, _Tp, _Allocator=_Allocator()`) -> `forward_list<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**list** (`_InputIterator, _InputIterator, _Allocator=_Allocator()`) -> `list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
**list** (`size_t, _Tp, _Allocator=_Allocator()`) -> `list<_Tp, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**map** (`_InputIterator, _InputIterator, _Allocator`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
**map** (`_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
**map** (`initializer_list<pair<_Key, _Tp>>, _Allocator`) -> `map<_Key, _Tp, less<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
**map** (`initializer_list<pair<_Key, _Tp>>, _Compare=_Compare(), _Allocator=_Allocator()`) -> `map<_Key, _Tp, _Compare, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**multimap** (`_InputIterator, _InputIterator, _Allocator`) -> `multimap<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
**multimap** (`_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()`) -> `multimap<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
**multimap** (`initializer_list<pair<_Key, _Tp>>, _Allocator`) -> `multimap<_Key, _Tp, less<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
**multimap** (`initializer_list<pair<_Key, _Tp>>, _Compare=_Compare(), _Allocator=_Allocator()`) -> `multimap<_Key, _Tp, _Compare, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
**multiset** (`_InputIterator, _InputIterator, _Allocator`) -> `multiset<typename iterator_traits<_InputIterator>::value_type, less<typename iterator_traits<_InputIterator>::value_type>, _Allocator>`

- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`_InputIterator`, `_InputIterator`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`initializer_list< _Key >`, `_Allocator`) -> `multiset< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**multiset** (`initializer_list< _Key >`, `_Compare=_Compare()`, `_Allocator=_Allocator()`) -> `multiset< _Key, _Compare, _Allocator >`
- `template<size_t _Nb>`  
**constexpr** `bitset< _Nb >` **operator&** (`const bitset< _Nb > &__x`, `const bitset< _Nb > &__y`) `noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
**std::basic\_ostream**< `_CharT`, `_Traits` > & **operator<<** (`std::basic_ostream< _CharT, _Traits > &__os`, `const bitset< _Nb > &__x`)
- `template<typename _Tp, typename _Alloc>`  
**constexpr** `__detail::__synth3way_t< _Tp >` **operator<=>** (`const deque< _Tp, _Alloc > &__x`, `const deque< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Alloc>`  
**constexpr** `__detail::__synth3way_t< _Tp >` **operator<=>** (`const forward_list< _Tp, _Alloc > &__x`, `const forward_list< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Alloc>`  
**constexpr** `__detail::__synth3way_t< _Tp >` **operator<=>** (`const list< _Tp, _Alloc > &__x`, `const list< _Tp, _Alloc > &__y`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > >` **operator<=>** (`const map< _Key, _Tp, _Compare, _Alloc > &__lhs`, `const map< _Key, _Tp, _Compare, _Alloc > &__rhs`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< pair< const _Key, _Tp > >` **operator<=>** (`const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs`, `const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs`)
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< _Key >` **operator<=>** (`const multiset< _Key, _Compare, _Alloc > &__lhs`, `const multiset< _Key, _Compare, _Alloc > &__rhs`)
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< _Key >` **operator<=>** (`const set< _Key, _Compare, _Alloc > &__lhs`, `const set< _Key, _Compare, _Alloc > &__rhs`)
- `template<typename _Tp, typename _Alloc>`  
**constexpr** `__detail::__synth3way_t< _Tp >` **operator<=>** (`const vector< _Tp, _Alloc > &__x`, `const vector< _Tp, _Alloc > &__y`)
- `template<typename _Tp, typename _Alloc>`  
**bool** **operator==** (`const deque< _Tp, _Alloc > &__lhs`, `const deque< _Tp, _Alloc > &__rhs`)
- `template<typename _Tp, typename _Alloc>`  
**bool** **operator==** (`const forward_list< _Tp, _Alloc > &__lx`, `const forward_list< _Tp, _Alloc > &__ly`)
- `template<typename _Tp, typename _Alloc>`  
**bool** **operator==** (`const list< _Tp, _Alloc > &__lhs`, `const list< _Tp, _Alloc > &__rhs`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
**bool** **operator==** (`const map< _Key, _Tp, _Compare, _Allocator > &__lhs`, `const map< _Key, _Tp, _Compare, _Allocator > &__rhs`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
**bool** **operator==** (`const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs`, `const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs`)

- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`constexpr bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Tp, typename _Alloc>`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Tp, typename _Alloc>`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Tp, typename _Alloc>`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator`  
`> &__rhs) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__`  
`y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*`  
`conditional */)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash,`  
`_Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp,`  
`_Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash,`  
`_Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, <`  
`_Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc>`  
`constexpr void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require<`  
`_Allocator<_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >,`  
`__iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator`  
`> >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require<`  
`_Allocator<_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator)`  
`-> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t<`  
`_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename =`  
`_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, <`  
`_Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash,`  
`equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<`  
`t<_InputIterator>>, typename _Allocator = allocator<__iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>,`  
`typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_<`  
`_Allocator>>>`  
`unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=<`  
`Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_<`  
`val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key`  
`>, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, <`  
`_Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>,`  
`typename = _RequireAllocator<_Allocator>>>`

- unordered\_map** (initializer\_list< pair< \_Key, \_Tp > >, typename unordered\_map< int, int >::size\_type, \_Hash, \_Allocator) -> unordered\_map< \_Key, \_Tp, \_Hash, equal\_to< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Hash = hash<\_Key>, typename \_Pred = equal\_to<\_Key>, typename \_Allocator = allocator<pair<const \_Key, \_Tp>>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
 **unordered\_map** (initializer\_list< pair< \_Key, \_Tp > >, typename unordered\_map< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, hash< \_\_iter\_key\_t< \_InputIterator > >, equal\_to< \_\_iter\_key\_t< \_InputIterator > >, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (\_InputIterator, \_InputIterator, unordered\_multimap< int, int >::size\_type, \_Allocator) -> unordered\_multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, hash< \_\_iter\_key\_t< \_InputIterator > >, equal\_to< \_\_iter\_key\_t< \_InputIterator > >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (\_InputIterator, \_InputIterator, unordered\_multimap< int, int >::size\_type, \_Hash, \_Allocator) -> unordered\_multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, \_Hash, equal\_to< \_\_iter\_key\_t< \_InputIterator > >, \_Allocator >
  - template<typename \_InputIterator, typename \_Hash = hash<\_\_iter\_key\_t<\_InputIterator>>, typename \_Pred = equal\_to<\_\_iter\_key\_t<\_InputIterator>>, typename \_Allocator = allocator<\_\_iter\_to\_alloc\_t<\_InputIterator>>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
 **unordered\_multimap** (\_InputIterator, \_InputIterator, unordered\_multimap< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, \_Hash, \_Pred, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, hash< \_Key >, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Hash, typename \_Allocator, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type, \_Hash, \_Allocator) -> unordered\_multimap< \_Key, \_Tp, \_Hash, equal\_to< \_Key >, \_Allocator >
  - template<typename \_Key, typename \_Tp, typename \_Hash = hash<\_Key>, typename \_Pred = equal\_to<\_Key>, typename \_Allocator = allocator<pair<const \_Key, \_Tp>>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireNotAllocator<\_Pred>, typename = \_RequireAllocator<\_Allocator>>>
 **unordered\_multimap** (initializer\_list< pair< \_Key, \_Tp > >, unordered\_multimap< int, int >::size\_type={}, \_Hash=\_Hash(), \_Pred=\_Pred(), \_Allocator=\_Allocator()) -> unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
  - template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>
 **unordered\_multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >



```

unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) ->
unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits<
_InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator
>

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -
> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename
iterator_traits< _InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred
= equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input
Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename
= _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(),
_Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator
>::value_type, _Hash, _Pred, _Allocator >

• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (initializer_list<_Tp >, _Allocator) -> unordered_multiset< _Tp, hash<_Tp >, equal_to<
_Tp >, _Allocator >

• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (initializer_list<_Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_
multiset< _Tp, hash<_Tp >, equal_to<_Tp >, _Allocator >

• template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (initializer_list<_Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) ->
unordered_multiset< _Tp, _Hash, equal_to<_Tp >, _Allocator >

• template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>
unordered_multiset (initializer_list<_Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(),
_Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>
unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits<_Input
Iterator >::value_type, hash< typename iterator_traits<_InputIterator >::value_type >, equal_to< typename
iterator_traits<_InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_
set< typename iterator_traits<_InputIterator >::value_type, hash< typename iterator_traits<_InputIterator
>::value_type >, equal_to< typename iterator_traits<_InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) ->
unordered_set< typename iterator_traits<_InputIterator >::value_type, _Hash, equal_to< typename
iterator_traits<_InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred
= equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input
Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename
= _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>
unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(),
_Allocator=_Allocator()) -> unordered_set< typename iterator_traits<_InputIterator >::value_type, _Hash, _Pred, _Allocator >

```



- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
`unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >`
- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
`vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >`

#### 4.11.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

#### 4.11.2 Function Documentation

##### swap()

```
template<typename _Tp, typename _Alloc>
void std::__debug::swap (
 forward_list< _Tp, _Alloc > & __lx,
 forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See `std::forward_list::swap()`.

## 4.12 std::\_\_detail Namespace Reference

### Classes

- struct [BracketMatcher](#)
- class [\\_Compiler](#)
- class [\\_Executor](#)
- struct [\\_List\\_node\\_base](#)
- struct [\\_List\\_node\\_header](#)
- struct [\\_Quoted\\_string](#)
- class [\\_Scanner](#)
- class [\\_StateSeq](#)

## Typedefs

- `template<typename _Tp, typename _Up>`  
using **\_\_cmp3way\_res\_t**
- `template<typename _Tp>`  
using **\_\_cref**
- `template<typename _Iter>`  
using **\_\_iter\_concept**
- `template<typename _Tp>`  
using **\_\_iter\_diff\_t**
- `template<typename _Iter, typename _Tp = _Iter>`  
using **\_\_iter\_traits**
- `template<typename _Tp>`  
using **\_\_iter\_value\_t**
- `template<typename _Tp>`  
using **\_\_range\_iter\_t**
- `template<typename _Tp>`  
using **\_\_unsigned\_least\_t**
- `template<typename _Tp>`  
using **\_\_with\_ref**
- `template<typename _CharT>`  
using **\_Matcher**
- `typedef long` **\_StateIdT**

## Enumerations

- `enum` **\_ExecutorFrameOpcode** : unsigned char {  
  **\_S\_fopcode\_next** , **\_S\_fopcode\_fallback\_next** , **\_S\_fopcode\_rep\_once\_more** , **\_S\_fopcode\_fallback\_↵**  
  **rep\_once\_more** ,  
  **\_S\_fopcode\_posix\_alternative** , **\_S\_fopcode\_merge\_sol** , **\_S\_fopcode\_restore\_cur\_results** , **\_S\_↵**  
  **fopcode\_restore\_rep\_count** ,  
  **\_S\_fopcode\_decrement\_rep\_count** }
- `enum` **\_Opcode** : int {  
  **\_S\_opcode\_unknown** , **\_S\_opcode\_alternative** , **\_S\_opcode\_repeat** , **\_S\_opcode\_backref** ,  
  **\_S\_opcode\_line\_begin\_assertion** , **\_S\_opcode\_line\_end\_assertion** , **\_S\_opcode\_word\_boundary** , **\_S\_↵**  
  **opcode\_subexpr\_lookahead** ,  
  **\_S\_opcode\_subexpr\_begin** , **\_S\_opcode\_subexpr\_end** , **\_S\_opcode\_dummy** , **\_S\_opcode\_match** ,  
  **\_S\_opcode\_accept** }
- `enum class` **\_RegexExecutorPolicy** : int { **\_S\_auto** , **\_S\_alternate** }

## Functions

- `template<typename _Res, typename _Tp>`  
constexpr **\_Res** **\_\_abs\_r** (\_Tp \_\_val)
- `template<typename>`  
void **\_\_abs\_r** (bool)=delete
- `template<typename _ChTraits>`  
constexpr auto **\_\_char\_traits\_cmp\_cat** (int \_\_cmp) noexcept
- `template<typename... _Ts>`  
constexpr auto **\_\_common\_cmp\_cat** ()
- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>`  
constexpr **\_Container::size\_type** **\_\_erase\_if** (\_Container &\_\_cont, \_UnsafeContainer &\_\_ucont, \_Predicate \_\_↵  
pred)

- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>`  
`_Container::size_type __erase_nodes_if (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)`
- `template<typename _ValT, typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & __extract_params (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)`
- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool __from_chars_alnum (const char * __first, const char * __last, _Tp & __val, int __base)`
- `template<bool _DecOnly = false>`  
`constexpr unsigned char __from_chars_alnum_to_val (unsigned char __c)`
- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool __from_chars_pow2_base (const char * __first, const char * __last, _Tp & __val, int __base)`
- `template<typename _Tp>`  
`constexpr _Tp __gcd (_Tp __m, _Tp __n)`
- `template<typename _Tp>`  
`_Tp * __get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _InputIterator, typename _RealType>`  
`bool __load_first2 (_InputIterator & __first, _InputIterator __last, _RealType * __out)`
- `template<typename _Tp>`  
`constexpr bool __p1_representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>`  
`constexpr bool __raise_and_add (_Tp & __val, int __base, unsigned char __c)`
- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT>`  
`bool __regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > & __m, const basic_regex< _CharT, _TraitsT > & __re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy __policy, bool __← match_mode)`
- `template<typename _Tp>`  
`constexpr bool __representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>`  
`void __return_temporary_buffer (_Tp * __p, size_t __len)`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars (char * __first, char * __last, _Tp __val, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_10 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr void __to_chars_10_impl (char * __first, unsigned __len, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_16 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_2 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to_chars_result __to_chars_8 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr unsigned __to_chars_len (_Tp __value, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr unsigned __to_chars_len_2 (_Tp __value) noexcept`
- `template<typename _CharT, typename _Traits, typename _String>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const _Quoted_string< _String, _CharT > & __str)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const _Quoted_string< const _CharT *, _CharT > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > & __str)`

## Variables

- `template<typename _Tp>`  
`constexpr unsigned __cmp_cat_id`
- `template<> constexpr unsigned __cmp_cat_id< partial_ordering >`
- `template<> constexpr unsigned __cmp_cat_id< strong_ordering >`
- `template<> constexpr unsigned __cmp_cat_id< weak_ordering >`
- `template<typename _Tp>`  
`constexpr bool __destructible`
- `template<typename _Tp>`  
`constexpr bool __destructible< _Tp & >`
- `template<typename _Tp>`  
`constexpr bool __destructible< _Tp && >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool __destructible< _Tp[_Nm]>`
- `template<typename _Tp>`  
`constexpr bool __destructible_impl`
- `template<typename _Tp>`  
`constexpr bool __destructible_impl< _Tp >`
- `template<typename _Tp>`  
`constexpr bool __is_ref_wrapper`
- `template<typename _Tp>`  
`constexpr bool __is_ref_wrapper< reference_wrapper< _Tp > >`
- `constexpr _StateIdT _S_invalid_state_id`

### 4.12.1 Detailed Description

Implementation details not part of the namespace `std` interface.

### 4.12.2 Function Documentation

#### `__from_chars_alnum()`

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_alnum (
 const char *& __first,
 const char * __last,
 _Tp & __val,
 int __base) [constexpr]
```

`std::from_chars` implementation for integers in any base. If `_DecOnly` is true, then we may assume `__base` is at most 10.

Referenced by [std::from\\_chars\(\)](#).

#### `__from_chars_pow2_base()`

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_pow2_base (
 const char *& __first,
 const char * __last,
 _Tp & __val,
 int __base) [constexpr]
```

`std::from_chars` implementation for integers in a power-of-two base. If `_DecOnly` is true, then we may assume `__base` is at most 8.

Referenced by [std::from\\_chars\(\)](#).

**operator<<()** [1/2]

```
template<typename _CharT, typename _Traits, typename _String>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const _Quoted_string< _String, _CharT > & __str)
```

Insertor for quoted strings.

**operator<<()** [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const _Quoted_string< const _CharT *, _CharT > & __str)
```

Insertor for quoted strings.

**operator>>()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str)
```

Extractor for delimited strings. The left and right delimiters can be different.

References [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::flags\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::good\(\)](#), [std::ios\\_base::setf\(\)](#), [std::ios\\_base::skipws](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**4.13 std::\_\_parallel Namespace Reference****Classes**

- struct [\\_CRandNumber](#)

**Functions**

- template<typename \_\_RAIter, typename \_Tp, typename \_BinaryOperation>  
\_Tp **accumulate\_switch** (\_\_RAIter \_\_begin, \_\_RAIter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter, typename \_Tp, typename \_BinaryOperation, typename \_IteratorTag>  
\_Tp **accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, \_IteratorTag)
- template<typename \_Iter, typename \_Tp, typename \_IteratorTag>  
\_Tp **accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_IteratorTag)
- template<typename \_Iter, typename \_Tp, typename \_BinaryOper, typename \_Tag>  
\_Tp **accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_BinaryOper, \_Tag)
- template<typename \_Iter, typename \_Tp, typename \_Tag>  
\_Tp **accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_Tag)
- template<typename \_RAIter, typename \_Tp, typename \_BinaryOper>  
\_Tp **accumulate\_switch** (\_RAIter, \_RAIter, \_Tp, \_BinaryOper, [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation, typename \_IteratorTag1, typename \_IteratorTag2>  
\_OutputIterator **adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, \_IteratorTag1, \_IteratorTag2)
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation>  
\_OutputIterator **adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::\\_\\_Parallelism](#) \_\_parallelism\_tag)

- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter __adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_Filter __adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _Filter, typename _IterTag>`  
`_Filter __adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _FIterator, typename _IteratorTag>`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate>`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _RAIter>`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _BiPredicate>`  
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`iterator\_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`iterator\_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool __equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool __equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2>`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>`  
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2,`  
`_BinaryPredicate __comp, random\_access\_iterator\_tag, _IteratorTag)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random\_access\_iterator\_tag, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag>`  
`_Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _Iter, typename _Function, typename _IterTag>`  
`_Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>`  
`_OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>`  
`_OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag>`  
`void __generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag>`  
`void __generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator>`  
`void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IterTag1, typename _IterTag2>`  
`_Tp __inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1,  
_BinaryFunction2 __binary_op2, _IterTag1, _IterTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>`  
`_Tp __inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,  
_Predicate __pred, _IteratorTag1, _IteratorTag2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAlter1, typename _RAlter2, typename _Predicate>`  
`bool __lexicographical_compare_switch (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _RAlter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAlter, typename _Compare>`  
`_RAlter __max_element_switch (_RAlter __begin, _RAlter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_Olter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare>`  
`_Olter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAlter, typename _Compare>`  
`_RAlter __min_element_switch (_RAlter __begin, _RAlter __end, _Compare __comp, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAlter1, typename _RAlter2, typename _Predicate>`  
`pair< _RAlter1, _RAlter2 > __mismatch_switch (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAlter1, typename _RAlter2, typename _Predicate>`  
`pair< _RAlter1, _RAlter2 > __mismatch_switch (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _RAlter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`



- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag>`  
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag>`  
`_Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag>`  
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag>`  
`void __replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag>`  
`void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag>`  
`void __replace_switch (_Filterator __begin, _Filterator __end, const _Tp & __old_value, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>`  
`_Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>`  
`_Filterator __search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp & __val, \_\_BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val, \_\_BinaryPredicate __binary_pred, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2>`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, \_\_BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2>`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, \_\_BinaryPredicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`

- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,  
_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,  
_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter __set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator __set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter __set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter`  
`__result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>`  
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation>`  
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>`  
`_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>`  
`_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>`  
`_RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>`  
`_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp, gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp accumulate (_Iter, _Iter, _Tp, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, gnu\_parallel::Parallelism)`

- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism  
\_\_parallelism\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation  
__binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp>`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp>`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value,  
\_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type count ( _Iter __begin, _Iter __end, const _Tp &__value,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type count_if ( _Iter __begin, _Iter __end, _Predicate __pred,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type count_if ( _Iter __begin, _Iter __end, _Predicate __pred,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Iter find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp>`  
`_Iter find ( _Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate`  
`__comp)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate`  
`__comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter>`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filter>`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`

- `template<typename _Iter, typename _Predicate>`  
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Iterator, typename _Function>`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iterator, typename _Function>`  
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Generator>`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Generator>`  
`void generate (_Filterator __begin, _Filterator __end, _Generator __gen)`
- `template<typename _Filterator, typename _Generator>`  
`void generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __↵  
parallelism_tag)`
- `template<typename _Filterator, typename _Generator>`  
`void generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism  
parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`

- `template<typename _Iter1, typename _Iter2>`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`__Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __Predicate`  
`__pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter>`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism`  
`__parallelism_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator max_element (_Filterator __begin, _Filterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵`  
`result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵`  
`result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵`  
`result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵`  
`result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`



- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter>`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end)`
- `template<typename _Iterator>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Compare>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end, _Compare __comp)`
- `template<typename _Iterator, typename _Compare>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Compare>`  
`_Iterator min_element (_Iterator __begin, _Iterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter>`  
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end)`
- `template<typename _RAlter>`  
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _Compare>`  
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`



- template<typename \_RAIter, typename \_Compare>  
void **nth\_element** (\_RAIter \_\_begin, \_RAIter \_\_nth, \_RAIter \_\_end, \_Compare \_\_comp, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_RAIter>  
void **partial\_sort** (\_RAIter \_\_begin, \_RAIter \_\_middle, \_RAIter \_\_end)
- template<typename \_RAIter>  
void **partial\_sort** (\_RAIter \_\_begin, \_RAIter \_\_middle, \_RAIter \_\_end, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_RAIter, typename \_Compare>  
void **partial\_sort** (\_RAIter \_\_begin, \_RAIter \_\_middle, \_RAIter \_\_end, \_Compare \_\_comp)
- template<typename \_RAIter, typename \_Compare>  
void **partial\_sort** (\_RAIter \_\_begin, \_RAIter \_\_middle, \_RAIter \_\_end, \_Compare \_\_comp, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter, typename \_OutputIterator>  
\_OutputIterator **partial\_sum** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result)
- template<typename \_Iter, typename \_OutputIterator>  
\_OutputIterator **partial\_sum** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation>  
\_OutputIterator **partial\_sum** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_bin\_op, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter, typename \_OutputIterator, typename \_BinaryOperation>  
\_OutputIterator **partial\_sum** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename \_Iter, typename \_OIter>  
\_OIter **partial\_sum** (\_Iter, \_Iter, \_OIter \_\_result)
- template<typename \_Iter, typename \_OIter>  
\_OIter **partial\_sum** (\_Iter, \_Iter, \_OIter, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Iter, typename \_OIter, typename \_BinaryOper>  
\_OIter **partial\_sum** (\_Iter, \_Iter, \_OIter, \_BinaryOper)
- template<typename \_Iter, typename \_OIter, typename \_BinaryOper>  
\_OIter **partial\_sum** (\_Iter, \_Iter, \_OIter, \_BinaryOper, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Filter, typename \_Predicate>  
\_Filter **partition** (\_Filter, \_Filter, \_Predicate)
- template<typename \_Filter, typename \_Predicate>  
\_Filter **partition** (\_Filter, \_Filter, \_Predicate, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Filterator, typename \_Predicate>  
\_Filterator **partition** (\_Filterator \_\_begin, \_Filterator \_\_end, \_Predicate \_\_pred)
- template<typename \_Filterator, typename \_Predicate>  
\_Filterator **partition** (\_Filterator \_\_begin, \_Filterator \_\_end, \_Predicate \_\_pred, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_RAIter>  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end)
- template<typename \_RAIter>  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &&\_\_rand)
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void **random\_shuffle** (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rand, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Filter, typename \_Tp>  
void **replace** (\_Filter, \_Filter, const \_Tp &, const \_Tp &)
- template<typename \_Filter, typename \_Tp>  
void **replace** (\_Filter, \_Filter, const \_Tp &, const \_Tp &, [\\_\\_gnu\\_parallel::Parallelism](#))
- template<typename \_Filter, typename \_Tp>  
void **replace** (\_Filter, \_Filter, const \_Tp &, const \_Tp &, [\\_\\_gnu\\_parallel::sequential\\_tag](#))
- template<typename \_Filterator, typename \_Tp>  
void **replace** (\_Filterator \_\_begin, \_Filterator \_\_end, const \_Tp &\_\_old\_value, const \_Tp &\_\_new\_value)

- `template<typename _Filterator, typename _Tp>`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp>`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp>`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate>`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2>`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2)`
- `template<typename _Filterator1, typename _Filterator2>`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate>`  
`constexpr _Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __`  
`end2, _BinaryPredicate __pred)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate>`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, __`  
`BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp>`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _Filterator, typename _Integer, typename _Tp>`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate`  
`__binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate>`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _RAIter, typename _Compare>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, \_\_gnu\_parallel::BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

#### 4.13.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

#### 4.13.2 Function Documentation

##### search()

```
template<typename _ForwardIterator, typename _Searcher>
_FowardIterator std::__parallel::search (
 _ForwardIterator __first,
 _ForwardIterator __last,
 const _Searcher & __searcher) [inline]
```

Search a sequence using a Searcher object.

##### Parameters

|                         |                     |
|-------------------------|---------------------|
| <code>__first</code>    | A forward iterator. |
| <code>__last</code>     | A forward iterator. |
| <code>__searcher</code> | A callable object.  |

##### Returns

```
__searcher(__first, __last).first
```

## 4.14 std::chrono Namespace Reference

### Classes

- class [duration](#)
- struct [duration\\_values](#)
- class [gps\\_clock](#)
- class [hh\\_mm\\_ss](#)

- struct [steady\\_clock](#)
- struct [system\\_clock](#)
- class [tai\\_clock](#)
- class [time\\_point](#)
- struct [treat\\_as\\_floating\\_point](#)
- class [tzdb\\_list](#)
- class [utc\\_clock](#)

### Typedefs

- using [days](#)
- using [file\\_clock](#)
- template<typename [\\_Duration](#)>  
using [file\\_time](#)
- using [gps\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [gps\\_time](#)
- using [high\\_resolution\\_clock](#)
- using [hours](#)
- using [local\\_days](#)
- using [local\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [local\\_time](#)
- using [microseconds](#)
- using [milliseconds](#)
- using [minutes](#)
- using [months](#)
- using [nanoseconds](#)
- using [seconds](#)
- using [sys\\_days](#)
- using [sys\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [sys\\_time](#)
- using [tai\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [tai\\_time](#)
- using [utc\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [utc\\_time](#)
- using [weeks](#)
- using [years](#)
- using [zoned\\_seconds](#)

### Enumerations

- enum class [choose](#) { [earliest](#) , [latest](#) }

## Functions

- `template<typename _Rep, typename _Period>`  
`constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > abs (duration< _Rep, _Period > __d)`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > ceil (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > ceil (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _DestClock, typename _SourceClock, typename _Duration>`  
`requires __detail::__clock_convs< _DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys< _DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc< _DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc< _DestClock, _SourceClock, _Duration>`  
`auto clock_cast (const time_point< _SourceClock, _Duration > &__t)`
- `const time_zone * current_zone ()`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > duration_cast (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > floor (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > floor (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, day & __d, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, duration< _Rep, _Period > & __d, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, file_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, gps_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, local_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, month & __m, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, month_day & __md, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT * __fmt, sys_time< _Duration > & __tp, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`



- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, weekday & __wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year & __y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year_month & __ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>>`  
`basic_istream< _CharT, _Traits > & from_stream (basic_istream< _CharT, _Traits > &__is, const _CharT *__`  
`__fmt, year_month_day & __ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__`  
`offset=nullptr)`
- `template<typename _Duration>`  
`leap_second_info get_leap_second_info (const utc_time< _Duration > &__ut)`
- `const tzdb & get_tzdb ()`
- `tzdb_list & get_tzdb_list ()`
- `constexpr bool is_am (const hours & __h) noexcept`
- `constexpr bool is_pm (const hours & __h) noexcept`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > local_time_format (local_time< _Duration > __time, const string *__`  
`__abbrev=nullptr, const seconds *__offset_sec=nullptr)`
- `const time_zone * locate_zone (string_view __tz_name)`
- `constexpr hours make12 (const hours & __h) noexcept`
- `constexpr hours make24 (const hours & __h, bool __is_pm) noexcept`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const day & __d)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const file_time<`  
`_Duration > & __t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const gps_time<`  
`_Duration > & __t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`hh_mm_ss< _Duration > & __hms)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info`  
`& __li)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`requires requires(const sys_time<_Duration>& __st) { __os << __st; }`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const local_time<`  
`_Duration > & __lt)`

- [illegible]

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const year_↵`  
`month_weekday_last &__ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const zoned_↵`  
`time< _Duration, _TimeZonePtr > &__t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`duration< _Rep, _Period > &__d)`
- `template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>`  
`bool operator== (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)`
- `template<typename _CharT, __detail::__parsable< _CharT > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable &__tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes`  
`&__offset)`
- `template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const _CharT * __fmt, _Parsable &__tp, minutes &__offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, ↵`  
`_Traits, _Alloc > &__abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵`  
`parsable< _CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, ↵`  
`_Traits, _Alloc > &__abbrev, minutes &__offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<↵`  
`_CharT, _Traits, _StrT, minutes > _Parsable>`  
`auto parse (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &__offset)`
- `const tldb & reload_tldb ()`
- `string remote_version ()`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr enable_if_t< __and< __is_duration< _ToDur >, __not< treat_as_floating_point< typename _To↵`  
`Dur::rep > >::value, _ToDur > round (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,↵`  
`time_point< _Clock, _ToDur > > round (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > time_point_cast`  
`(const time_point< _Clock, _Dur > &__t)`
- `zoned_time () -> zoned_time< seconds >`
- `template<typename _TimeZonePtrOrName>`  
`zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _Time↵`  
`ZonePtrOrName > >`
- `template<typename _TimeZonePtrOrName, typename _Duration>`  
`zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time<`  
`common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > > >`

- `template<typename _TimeZonePtrOrName, typename _Duration>`  
**zoned\_time** (`_TimeZonePtrOrName &&`, `sys_time<_Duration>`) -> `zoned_time<common_type_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>`  
**zoned\_time** (`_TimeZonePtrOrName &&`, `zoned_time<_Duration, _TimeZonePtr2>`, `choose=choose::earliest`) -> `zoned_time<common_type_t<_Duration, seconds>, __time_zone_representation<_TimeZonePtrOrName> >`
- `template<typename _Duration>`  
**zoned\_time** (`sys_time<_Duration>`) -> `zoned_time<common_type_t<_Duration, seconds> >`
- `template<typename _Clock, typename _Dur1, three_way_comparable_with<_Dur1> _Dur2>`  
`constexpr auto operator<=>` (`const time_point<_Clock, _Dur1> &__lhs`, `const time_point<_Clock, _Dur2> &__rhs`)
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator<` (`const time_point<_Clock, _Dur1> &__lhs`, `const time_point<_Clock, _Dur2> &__rhs`)
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator<=` (`const time_point<_Clock, _Dur1> &__lhs`, `const time_point<_Clock, _Dur2> &__rhs`)
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator>` (`const time_point<_Clock, _Dur1> &__lhs`, `const time_point<_Clock, _Dur2> &__rhs`)
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool operator>=` (`const time_point<_Clock, _Dur1> &__lhs`, `const time_point<_Clock, _Dur2> &__rhs`)

## Variables

- `constexpr month` **April**
- `constexpr month` **August**
- `constexpr month` **December**
- `constexpr month` **February**
- `constexpr weekday` **Friday**
- `template<typename _Tp>`  
`constexpr bool is_clock_v`
- `template<typename _Tp>`  
`constexpr bool is_clock_v<_Tp>`
- `template<>` `constexpr bool is_clock_v<file_clock>`
- `template<>` `constexpr bool is_clock_v<gps_clock>`
- `template<>` `constexpr bool is_clock_v<steady_clock>`
- `template<>` `constexpr bool is_clock_v<system_clock>`
- `template<>` `constexpr bool is_clock_v<tai_clock>`
- `template<>` `constexpr bool is_clock_v<utc_clock>`
- `constexpr month` **January**
- `constexpr month` **July**
- `constexpr month` **June**
- `constexpr last_spec` **last**
- `constexpr month` **March**
- `constexpr month` **May**
- `constexpr weekday` **Monday**
- `constexpr month` **November**
- `constexpr month` **October**

- constexpr weekday **Saturday**
- constexpr month **September**
- constexpr weekday **Sunday**
- constexpr weekday **Thursday**
- template<typename \_Rep>  
constexpr bool **treat\_as\_floating\_point\_v**
- template<> constexpr bool **treat\_as\_floating\_point\_v**< double >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< float >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< int >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< long >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< long double >
- template<> constexpr bool **treat\_as\_floating\_point\_v**< long long >
- constexpr weekday **Tuesday**
- constexpr weekday **Wednesday**
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< **duration**< \_Rep1, \_Period1 >, **duration**< \_Rep2, \_Period2 > >::type **operator-** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< **duration**< \_Rep1, \_Period1 >, **duration**< \_Rep2, \_Period2 > >::type **operator+** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Rep2, typename \_Period>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep2, \_Rep1 >, \_Period > **operator\*** (const \_Rep1 &\_\_s, const **duration**< \_Rep2, \_Period > &\_\_d)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 > >, \_Period > **operator/** (const **duration**< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< \_Rep1, \_Rep2 >::type **operator/** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep1, \_\_disable\_if\_is\_duration< \_Rep2 > >, \_Period > **operator%** (const **duration**< \_Rep1, \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr **common\_type**< **duration**< \_Rep1, \_Period1 >, **duration**< \_Rep2, \_Period2 > >::type **operator%** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr **duration**< \_\_common\_rep\_t< \_Rep1, \_Rep2 >, \_Period > **operator\*** (const **duration**< \_Rep1, \_\_enable\_if\_is\_duration< \_Period > &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool **operator<** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr auto **operator<=>** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool **operator<=** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool **operator>** (const **duration**< \_Rep1, \_Period1 > &\_\_lhs, const **duration**< \_Rep2, \_Period2 > &\_\_rhs)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 >`  
`&__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >`  
`operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const`  
`time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

#### 4.14.1 Detailed Description

ISO C++ 2011 namespace for date and time utilities.

### 4.15 `std::decimal` Namespace Reference

#### Classes

- class `decimal128`
- class `decimal32`
- class `decimal64`

#### Functions

- double `decimal128_to_double` (`decimal128 __d`)
- float `decimal128_to_float` (`decimal128 __d`)
- long double `decimal128_to_long_double` (`decimal128 __d`)
- long long `decimal128_to_long_long` (`decimal128 __d`)
- double `decimal32_to_double` (`decimal32 __d`)
- float `decimal32_to_float` (`decimal32 __d`)
- long double `decimal32_to_long_double` (`decimal32 __d`)
- long long `decimal32_to_long_long` (`decimal32 __d`)
- double `decimal64_to_double` (`decimal64 __d`)
- float `decimal64_to_float` (`decimal64 __d`)
- long double `decimal64_to_long_double` (`decimal64 __d`)
- long long `decimal64_to_long_long` (`decimal64 __d`)
- double `decimal_to_double` (`decimal128 __d`)
- double `decimal_to_double` (`decimal32 __d`)
- double `decimal_to_double` (`decimal64 __d`)
- float `decimal_to_float` (`decimal128 __d`)
- float `decimal_to_float` (`decimal32 __d`)
- float `decimal_to_float` (`decimal64 __d`)
- long double `decimal_to_long_double` (`decimal128 __d`)

- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)



- bool **operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)



- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator+](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, int \_\_rhs)

- **decimal128 operator-** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator-** ([decimal128](#) \_\_rhs)
- **decimal128 operator-** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, int \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator-** ([decimal32](#) \_\_rhs)
- **decimal128 operator-** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, int \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator-** ([decimal64](#) \_\_rhs)
- **decimal128 operator-** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator-** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator-** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator-** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)

- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, int \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator/** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, int \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, long long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator/** ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (long long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- **decimal128 operator/** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- **decimal32 operator/** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- **decimal64 operator/** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, int \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, long long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- **bool operator<** ([decimal32](#) \_\_lhs, int \_\_rhs)

- bool **operator**< (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal128 \_\_rhs)

- bool **operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned int \_\_rhs)

- bool **operator**> (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**> (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**> (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**>= (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal128 \_\_rhs)

- bool **operator**>= (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**>= (unsigned long long \_\_lhs, decimal64 \_\_rhs)

#### 4.15.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

#### 4.15.2 Function Documentation

##### decimal32\_to\_long\_long()

```
long long std::experimental::decimal32_to_long_long (
 decimal32 __d)
```

Non-conforming extension: Conversion to integral type.

## 4.16 std::experimental Namespace Reference

### Classes

- class [any](#)
- class [bad\\_any\\_cast](#)
- class [bad\\_optional\\_access](#)
- class [basic\\_string\\_view](#)
- struct [in\\_place\\_t](#)
- struct [nullopt\\_t](#)
- class [optional](#)
- class [ostream\\_joiner](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)
- class [propagate\\_const](#)

### Typedefs

- template<typename \_RAIter, typename \_Hash, typename \_Pred, typename \_Val = typename iterator\_traits<\_RAIter>::value\_type, typename \_Diff = typename iterator\_traits<\_RAIter>::difference\_type>  
using [\\_\\_boyer\\_moore\\_base\\_t](#)
- template<typename \_Tp>  
using [\\_\\_propagate\\_const\\_elem\\_type](#)
- using [erased\\_type](#)
- using [string\\_view](#)
- using [u16string\\_view](#)
- using [u32string\\_view](#)
- using [wstring\\_view](#)



## Functions

- `template<typename _Fn, typename _Tuple, std::size_t... _Idx>`  
`constexpr decltype(auto) __apply_impl (_Fn &&__f, _Tuple &&__t, std::index\_sequence< _Idx... >)`
- `template<typename _Tp, size_t _Nm, size_t... _Idx>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > __to_array (_Tp(&__a)[_Nm], index\_sequence< _Idx... >)`
- `std::default_random_engine & _S_randint_engine ()`
- `template<typename _ValueType>`  
`_ValueType any_cast (const any &__any)`
- `template<typename _Fn, typename _Tuple>`  
`constexpr decltype(auto) apply (_Fn &&__f, _Tuple &&__t)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory\_order __success, memory\_order __failure)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp>`  
`bool atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory\_order __success, memory\_order __failure)`
- `template<typename _Tp>`  
`void atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_exchange_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory\_order __mo)`
- `template<typename _Tp>`  
`bool atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > *__p, memory\_order __mo)`
- `template<typename _Tp>`  
`void atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > atomic_store_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory\_order __mo)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>`  
`void erase (basic\_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (deque< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void erase (vector< _Tp, _Alloc > &__cont, const _Up &__value)`



- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>`  
`void erase_if (basic\_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (deque< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (list< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void erase_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void erase_if (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void erase_if (vector< _Tp, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Mn, typename _Nn>`  
`constexpr common\_type\_t< _Mn, _Nn > gcd (_Mn __m, _Nn __n) noexcept`
- `template<typename _Del, typename _Tp>`  
`_Del * get\_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp>`  
`constexpr const _Tp & get\_underlying (const propagate\_const< _Tp > &__pt) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp & get\_underlying (propagate\_const< _Tp > &__pt) noexcept`
- `template<typename _Mn, typename _Nn>`  
`constexpr common\_type\_t< _Mn, _Nn > lcm (_Mn __m, _Nn __n)`
- `template<typename _Dest = void, typename... _Types>`  
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> make\_array (↵  
_Types &&... __t)`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↵  
Predicate = equal_to<>>>`  
`boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > make\_boyer\_moore\_horspool\_searcher (  
_RAlter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↵  
Predicate = equal_to<>>>`  
`boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > make\_boyer\_moore\_searcher (_RAIter __pat_↵  
first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>>`  
`default_searcher< _ForwardIterator, _BinaryPredicate > make\_default\_searcher (_ForwardIterator __pat_first,  
_FowardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Tp>`  
`constexpr observer_ptr< _Tp > make\_observer (_Tp *__p) noexcept`

- `template<typename _CharT, typename _Traits, typename _DelimT>`  
`ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > make_ostream_joiner (basic_ostream< _CharT, _Traits > &__os, _DelimT &&__delimiter)`
- `template<typename _Fn>`  
`auto not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator!= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool operator!= (const propagate_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator!= (nullptr_t, const propagate_const< _Tp > &__pu)`
- `template<typename _Tp>`  
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`bool operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `constexpr basic_string_view< char > operator""sv (const char *__str, size_t __len) noexcept`
- `constexpr basic_string_view< char16_t > operator""sv (const char16_t *__str, size_t __len) noexcept`
- `constexpr basic_string_view< char32_t > operator""sv (const char32_t *__str, size_t __len) noexcept`
- `constexpr basic_string_view< wchar_t > operator""sv (const wchar_t *__str, size_t __len) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator< (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const propagate_const< _Tp > &__pt, const _Up &__u)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)`
- `template<typename _Ch, typename _Tr, typename _Tp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator<= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator== (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, const _Up &__u)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool operator== (const propagate_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (nullptr_t, const propagate_const< _Tp > &__pu)`
- `template<typename _Tp>`  
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`constexpr bool operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator> (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool operator>= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const _Tp &__t, const propagate_const< _Up > &__pu)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`bool operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _IntType>`  
`_IntType randint (_IntType __a, _IntType __b)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `void reseed ()`
- `void reseed (default\_random\_engine::result\_type __value)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>`  
`_SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, ←  
Distance __n)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>`  
`_SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance  
__n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator>`  
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `void swap (any &__x, any &__y) noexcept`
- `template<typename _Tp>`  
`constexpr void swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`
- `template<typename _Tp>`  
`constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > swap (propagate\_const< _Tp > &__pt,  
propagate\_const< _Tp > &__pt2) noexcept(!__is_nothrow_swappable< _Tp >::value)`
- `template<typename _Tp>`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp>`  
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`constexpr array< remove\_cv\_t< _Tp >, _Nm > to_array (_Tp(&__a)[_Nm]) noexcept(is\_nothrow\_constructible<  
remove\_cv\_t< _Tp >, _Tp & >::value)`
- `template<typename _ValueType>`  
`_ValueType any_cast (any &__any)`
- `template<typename _ValueType, typename enable\_if<!__is_move_constructible< _ValueType >::value||is\_lvalue\_reference< _ValueType  
>::value, bool >::type = true>  
_ValueType any_cast (any &&__any)`
- `template<typename _ValueType, typename enable\_if<__is_move_constructible< _ValueType >::value &&!is\_lvalue\_reference< _ValueType  
>::value, bool >::type = false>  
_ValueType any_cast (any &&__any)`

- `template<typename _ValueType>`  
`const _ValueType * any\_cast (const any * __any) noexcept`
- `template<typename _ValueType>`  
`_ValueType * any\_cast (any * __any) noexcept`

## Variables

- `template<typename _Yp, typename _Tp>`  
`constexpr bool \_\_sp\_compatible\_v`
- `template<typename _Tp, typename _Yp>`  
`constexpr bool \_\_sp\_is\_constructible\_v`
- `constexpr in\_place\_t in\_place`
- `template<typename _Tp>`  
`constexpr bool is\_bind\_expression\_v`
- `template<typename _Tp>`  
`constexpr bool is\_error\_code\_enum\_v`
- `template<typename _Tp>`  
`constexpr bool is\_error\_condition\_enum\_v`
- `template<typename _Tp>`  
`constexpr int is\_placeholder\_v`
- `constexpr nullopt\_t nullopt`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_equal\_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_greater\_equal\_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_greater\_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_less\_equal\_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_less\_v`
- `template<typename _R1, typename _R2>`  
`constexpr bool ratio\_not\_equal\_v`
- `template<typename _Tp>`  
`constexpr size_t tuple\_size\_v`
- `template<typename _Tp>`  
`constexpr bool is\_void\_v`
- `template<typename _Tp>`  
`constexpr bool is\_null\_pointer\_v`
- `template<typename _Tp>`  
`constexpr bool is\_integral\_v`
- `template<typename _Tp>`  
`constexpr bool is\_floating\_point\_v`
- `template<typename _Tp>`  
`constexpr bool is\_array\_v`
- `template<typename _Tp>`  
`constexpr bool is\_pointer\_v`
- `template<typename _Tp>`  
`constexpr bool is\_lvalue\_reference\_v`

- `template<typename _Tp>`  
`constexpr bool is_rvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_enum_v`
- `template<typename _Tp>`  
`constexpr bool is_union_v`
- `template<typename _Tp>`  
`constexpr bool is_class_v`
- `template<typename _Tp>`  
`constexpr bool is_function_v`
- `template<typename _Tp>`  
`constexpr bool is_reference_v`
- `template<typename _Tp>`  
`constexpr bool is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool is_fundamental_v`
- `template<typename _Tp>`  
`constexpr bool is_object_v`
- `template<typename _Tp>`  
`constexpr bool is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool is_compound_v`
- `template<typename _Tp>`  
`constexpr bool is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool is_const_v`
- `template<typename _Tp>`  
`constexpr bool is_volatile_v`
- `template<typename _Tp>`  
`constexpr bool is_trivial_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool is_pod_v`
- `template<typename _Tp>`  
`constexpr bool is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool is_empty_v`
- `template<typename _Tp>`  
`constexpr bool is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool is_final_v`
- `template<typename _Tp>`  
`constexpr bool is_signed_v`

- `template<typename _Tp>`  
`constexpr bool is_unsigned_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_trivially_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_trivially_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_trivially_destructible_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool is_nothrow_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool has_virtual_destructor_v`



- `template<typename _Tp>`  
`constexpr size_t alignment_of_v`
  - `template<typename _Tp>`  
`constexpr size_t rank_v`
  - `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t extent_v`
  - `template<typename _Tp, typename _Up>`  
`constexpr bool is_same_v`
  - `template<typename _Tp>`  
`constexpr bool is_same_v< _Tp, _Tp >`
  - `template<typename _Base, typename _Derived>`  
`constexpr bool is_base_of_v`
  - `template<typename _From, typename _To>`  
`constexpr bool is_convertible_v`
- 
- `template<typename... _Bn>`  
`constexpr bool conjunction_v`
  - `template<typename... _Bn>`  
`constexpr bool disjunction_v`
  - `template<typename _Pp>`  
`constexpr bool negation_v`
- 
- `template<typename...>`  
`using void\_t`
  - `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using detected\_or`
  - `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using detected\_or\_t`
  - `template<template< typename... > class _Op, typename... _Args>`  
`using detected\_t`
  - `template<template< typename... > class _Op, typename... _Args>`  
`using is\_detected`
  - `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using is\_detected\_exact`
  - `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using is\_detected\_convertible`
  - `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool is\_detected\_v`
  - `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool is\_detected\_exact\_v`
  - `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool is\_detected\_convertible\_v`

#### 4.16.1 Detailed Description

Namespace for features defined in ISO Technical Specifications.

## 4.16.2 Function Documentation

### **gcd()**

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::gcd (
 _Mn __m,
 _Nn __n) [constexpr], [noexcept]
```

Greatest common divisor.

### **get\_deleter()**

```
template<typename _Del, typename _Tp>
_Del * std::experimental::fundamentals_v2::get_deleter (
 const shared_ptr< _Tp > & __p) [inline], [noexcept]
```

C++14 20.8.2.2.10.

### **lcm()**

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::lcm (
 _Mn __m,
 _Nn __n) [constexpr]
```

Least common multiple.

### **make\_boyer\_moore\_horspool\_searcher()**

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>,
typename _BinaryPredicate = equal_to<>>
boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1::make_boyer_moore_horspool_searcher (
 _RAIter __pat_first,
 _RAIter __pat_last,
 _Hash __hf = _Hash(),
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for boyer\_moore\_horspool\_searcher.

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

### **make\_boyer\_moore\_searcher()**

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>,
typename _BinaryPredicate = equal_to<>>
boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1::make_boyer_moore_searcher (
 _RAIter __pat_first,
 _RAIter __pat_last,
 _Hash __hf = _Hash(),
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for boyer\_moore\_searcher.

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

### **make\_default\_searcher()**

```
template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::fundamentals_v1::make_default_searcher (
```

```

 _ForwardIterator __pat_first,
 _ForwardIterator __pat_last,
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]

```

Generator function for default\_searcher.

### make\_ostream\_joiner()

```

template<typename _CharT, typename _Traits, typename _DelimT>
ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::fundamentals_v2::make_ostream_joiner (
 basic_ostream< _CharT, _Traits > & __os,
 _DelimT && __delimiter) [inline]

```

Object generator for ostream\_joiner.

References [std::forward\(\)](#).

### not\_fn()

```

template<typename _Fn>
auto std::experimental::fundamentals_v2::not_fn (
 _Fn && __fn) [inline], [noexcept]

```

[func.not\_fn] Function template not\_fn

References [std::forward\(\)](#).

### sample()

```

template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
_SampleIterator std::experimental::fundamentals_v2::sample (
 _PopulationIterator __first,
 _PopulationIterator __last,
 _SampleIterator __out,
 _Distance __n,
 _UniformRandomNumberGenerator && __g)

```

Take a random sample from a population.

References [std::forward\(\)](#).

## 4.16.3 Variable Documentation

### is\_bind\_expression\_v

```

template<typename _Tp>
bool std::experimental::fundamentals_v1::is_bind_expression_v [constexpr]

```

Variable template for std::is\_bind\_expression.

### is\_placeholder\_v

```

template<typename _Tp>
int std::experimental::fundamentals_v1::is_placeholder_v [constexpr]

```

Variable template for std::is\_placeholder.

## 4.17 std::filesystem Namespace Reference

### Classes

- class [directory\\_entry](#)
- class [directory\\_iterator](#)

- class `file_status`
- class `filesystem_error`
- class `path`
- class `recursive_directory_iterator`
- struct `space_info`

## Typedefs

- using `file_time_type`

## Enumerations

- enum class `copy_options` : unsigned short { `none` , `skip_existing` , `overwrite_existing` , `update_existing` , `recursive` , `copy_symlinks` , `skip_symlinks` , `directories_only` , `create_symlinks` , `create_hard_links` }
- enum class `directory_options` : unsigned char { `none` , `follow_directory_symlink` , `skip_permission_denied` }
- enum class `file_type` : signed char { `none` , `not_found` , `regular` , `directory` , `symlink` , `block` , `character` , `fifo` , `socket` , `unknown` }
- enum class `perm_options` : unsigned { `replace` , `add` , `remove` , `nofollow` }
- enum class `perms` : unsigned { `none` , `owner_read` , `owner_write` , `owner_exec` , `owner_all` , `group_read` , `group_write` , `group_exec` , `group_all` , `others_read` , `others_write` , `others_exec` , `others_all` , `all` , `set_uid` , `set_gid` , `sticky_bit` , `mask` , `unknown` }

## Functions

- `path absolute` (const `path` &\_\_p)
- `path absolute` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path canonical` (const `path` &\_\_p)
- `path canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- void `copy` (const `path` &\_\_from, const `path` &\_\_to)
- void `copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options)
- void `copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options, `error_code` &\_\_ec)
- void `copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec)
- bool `copy_file` (const `path` &\_\_from, const `path` &\_\_to)
- bool `copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option)
- bool `copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option, `error_code` &\_\_ec)
- bool `copy_file` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec)
- void `copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_symlink)
- void `copy_symlink` (const `path` &\_\_existing\_symlink, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept
- bool `create_directories` (const `path` &\_\_p)
- bool `create_directories` (const `path` &\_\_p, `error_code` &\_\_ec)
- bool `create_directory` (const `path` &\_\_p)
- bool `create_directory` (const `path` &\_\_p, const `path` &\_\_attributes)
- bool `create_directory` (const `path` &\_\_p, const `path` &\_\_attributes, `error_code` &\_\_ec) noexcept
- bool `create_directory` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- void `create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink)
- void `create_directory_symlink` (const `path` &\_\_to, const `path` &\_\_new\_symlink, `error_code` &\_\_ec) noexcept

- void **create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link)
- void **create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link, [error\\_code](#) &\_\_ec) noexcept
- void **create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **current\_path** ()
- void **current\_path** (const [path](#) &\_\_p)
- void **current\_path** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **current\_path** ([error\\_code](#) &\_\_ec)
- bool **equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool **equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noexcept
- bool **exists** (const [path](#) &\_\_p)
- bool **exists** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **exists** ([file\\_status](#)) noexcept
- uintmax\_t **file\_size** (const [path](#) &\_\_p)
- uintmax\_t **file\_size** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- uintmax\_t **hard\_link\_count** (const [path](#) &\_\_p)
- uintmax\_t **hard\_link\_count** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- size\_t **hash\_value** (const [path](#) &\_\_p) noexcept
- bool **is\_block\_file** (const [path](#) &\_\_p)
- bool **is\_block\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_block\_file** ([file\\_status](#) \_\_s) noexcept
- bool **is\_character\_file** (const [path](#) &\_\_p)
- bool **is\_character\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_character\_file** ([file\\_status](#) \_\_s) noexcept
- bool **is\_directory** (const [path](#) &\_\_p)
- bool **is\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_directory** ([file\\_status](#) \_\_s) noexcept
- bool **is\_empty** (const [path](#) &\_\_p)
- bool **is\_empty** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_fifo** (const [path](#) &\_\_p)
- bool **is\_fifo** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_fifo** ([file\\_status](#) \_\_s) noexcept
- bool **is\_other** (const [path](#) &\_\_p)
- bool **is\_other** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_other** ([file\\_status](#)) noexcept
- bool **is\_regular\_file** (const [path](#) &\_\_p)
- bool **is\_regular\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_regular\_file** ([file\\_status](#)) noexcept
- bool **is\_socket** (const [path](#) &\_\_p)
- bool **is\_socket** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_socket** ([file\\_status](#) \_\_s) noexcept
- bool **is\_symlink** (const [path](#) &\_\_p)
- bool **is\_symlink** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **is\_symlink** ([file\\_status](#)) noexcept
- [file\\_time\\_type](#) **last\_write\_time** (const [path](#) &\_\_p)
- [file\\_time\\_type](#) **last\_write\_time** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void **last\_write\_time** (const [path](#) &\_\_p, [file\\_time\\_type](#) \_\_new\_time)
- void **last\_write\_time** (const [path](#) &\_\_p, [file\\_time\\_type](#) \_\_new\_time, [error\\_code](#) &\_\_ec) noexcept
- [copy\\_options](#) & **operator&=** ([copy\\_options](#) &\_\_x, [copy\\_options](#) \_\_y) noexcept
- constexpr [copy\\_options](#) **operator^** ([copy\\_options](#) \_\_x, [copy\\_options](#) \_\_y) noexcept
- [copy\\_options](#) & **operator^=** ([copy\\_options](#) &\_\_x, [copy\\_options](#) \_\_y) noexcept

- constexpr **copy\_options operator|** (**copy\_options** \_\_x, **copy\_options** \_\_y) noexcept
  - **copy\_options & operator|=** (**copy\_options** &\_\_x, **copy\_options** \_\_y) noexcept
  - constexpr **copy\_options operator~** (**copy\_options** \_\_x) noexcept
  - void **permissions** (const **path** &, **perms**, **perm\_options**, **error\_code** &) noexcept
  - void **permissions** (const **path** &\_\_p, **perms** \_\_prms, **error\_code** &\_\_ec) noexcept
  - void **permissions** (const **path** &\_\_p, **perms** \_\_prms, **perm\_options** \_\_opts=perm\_options::replace)
  - **path proximate** (const **path** &\_\_p, const **path** &\_\_base, **error\_code** &\_\_ec)
  - **path proximate** (const **path** &\_\_p, const **path** &\_\_base=current\_path())
  - **path proximate** (const **path** &\_\_p, **error\_code** &\_\_ec)
  - **path read\_symlink** (const **path** &\_\_p)
  - **path read\_symlink** (const **path** &\_\_p, **error\_code** &\_\_ec)
  - **path relative** (const **path** &\_\_p, const **path** &\_\_base, **error\_code** &\_\_ec)
  - **path relative** (const **path** &\_\_p, const **path** &\_\_base=current\_path())
  - **path relative** (const **path** &\_\_p, **error\_code** &\_\_ec)
  - bool **remove** (const **path** &, **error\_code** &) noexcept
  - bool **remove** (const **path** &\_\_p)
  - uintmax\_t **remove\_all** (const **path** &)
  - uintmax\_t **remove\_all** (const **path** &, **error\_code** &)
  - void **rename** (const **path** &\_\_from, const **path** &\_\_to)
  - void **rename** (const **path** &\_\_from, const **path** &\_\_to, **error\_code** &\_\_ec) noexcept
  - void **resize\_file** (const **path** &\_\_p, uintmax\_t \_\_size)
  - void **resize\_file** (const **path** &\_\_p, uintmax\_t \_\_size, **error\_code** &\_\_ec) noexcept
  - **space\_info space** (const **path** &\_\_p)
  - **space\_info space** (const **path** &\_\_p, **error\_code** &\_\_ec) noexcept
  - **file\_status status** (const **path** &)
  - **file\_status status** (const **path** &, **error\_code** &) noexcept
  - bool **status\_known** (**file\_status**) noexcept
  - void **swap** (**path** &\_\_lhs, **path** &\_\_rhs) noexcept
  - **file\_status symlink\_status** (const **path** &)
  - **file\_status symlink\_status** (const **path** &, **error\_code** &) noexcept
  - **path temp\_directory\_path** ()
  - **path temp\_directory\_path** (**error\_code** &\_\_ec)
  - template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>, typename \_CharT = \_\_detail::\_value\_type\_is\_char\_or\_char8\_t<\_InputIterator>>  
**path u8path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>, typename \_CharT = \_\_detail::\_value\_type\_is\_char\_or\_char8\_t<\_Source>>  
**path u8path** (const \_Source &\_\_source)
  - **path weakly\_canonical** (const **path** &\_\_p)
  - **path weakly\_canonical** (const **path** &\_\_p, **error\_code** &\_\_ec)
- 
- constexpr **perms operator|** (**perms** \_\_x, **perms** \_\_y) noexcept
  - constexpr **perms operator^** (**perms** \_\_x, **perms** \_\_y) noexcept
  - constexpr **perms operator~** (**perms** \_\_x) noexcept
  - **perms & operator&=** (**perms** &\_\_x, **perms** \_\_y) noexcept
  - **perms & operator|=** (**perms** &\_\_x, **perms** \_\_y) noexcept
  - **perms & operator^=** (**perms** &\_\_x, **perms** \_\_y) noexcept
- 
- constexpr **perm\_options operator|** (**perm\_options** \_\_x, **perm\_options** \_\_y) noexcept

- constexpr [perm\\_options](#) **operator**<sup>^</sup> ([perm\\_options](#) \_\_x, [perm\\_options](#) \_\_y) noexcept
- constexpr [perm\\_options](#) **operator**<sup>~</sup> ([perm\\_options](#) \_\_x) noexcept
- [perm\\_options](#) & **operator**&= ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
- [perm\\_options](#) & **operator**|= ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
- [perm\\_options](#) & **operator**<sup>^</sup>= ([perm\\_options](#) &\_\_x, [perm\\_options](#) \_\_y) noexcept
- constexpr [directory\\_options](#) **operator**| ([directory\\_options](#) \_\_x, [directory\\_options](#) \_\_y) noexcept
- constexpr [directory\\_options](#) **operator**<sup>^</sup> ([directory\\_options](#) \_\_x, [directory\\_options](#) \_\_y) noexcept
- constexpr [directory\\_options](#) **operator**<sup>~</sup> ([directory\\_options](#) \_\_x) noexcept
- [directory\\_options](#) & **operator**&= ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept
- [directory\\_options](#) & **operator**|= ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept
- [directory\\_options](#) & **operator**<sup>^</sup>= ([directory\\_options](#) &\_\_x, [directory\\_options](#) \_\_y) noexcept

#### 4.17.1 Detailed Description

ISO C++ 2017 namespace for File System library.

## 4.18 std::literals Namespace Reference

### Namespaces

- namespace [chrono\\_literals](#)

### Functions

- constexpr chrono::day **operator**""d (unsigned long long \_\_d) noexcept
- template<char... \_Digits>  
constexpr chrono::hours **operator**""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > **operator**""h (long double \_\_hours)
- template<char... \_Digits>  
constexpr chrono::minutes **operator**""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > **operator**""min (long double \_\_mins)
- template<char... \_Digits>  
constexpr chrono::milliseconds **operator**""ms ()
- constexpr chrono::duration< long double, milli > **operator**""ms (long double \_\_msecs)
- template<char... \_Digits>  
constexpr chrono::nanoseconds **operator**""ns ()
- constexpr chrono::duration< long double, nano > **operator**""ns (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr chrono::seconds **operator**""s ()
- constexpr chrono::duration< long double > **operator**""s (long double \_\_secs)
- constexpr [basic\\_string\\_view](#)< char > **operator**""sv (const char \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char16\_t > **operator**""sv (const char16\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char32\_t > **operator**""sv (const char32\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< wchar\_t > **operator**""sv (const wchar\_t \*\_\_str, size\_t \_\_len) noexcept
- template<char... \_Digits>  
constexpr chrono::microseconds **operator**""us ()
- constexpr chrono::duration< long double, micro > **operator**""us (long double \_\_usecs)
- constexpr chrono::year **operator**""y (unsigned long long \_\_y) noexcept

#### 4.18.1 Detailed Description

ISO C++ inline namespace for literal suffixes.

### 4.19 `std::literals::chrono_literals` Namespace Reference

#### Functions

- constexpr `chrono::day` `operator""d` (unsigned long long \_\_d) noexcept
- template<char... \_Digits>  
constexpr `chrono::hours` `operator""h` ()
- constexpr `chrono::duration`< long double, `ratio`< 3600, 1 > > `operator""h` (long double \_\_hours)
- template<char... \_Digits>  
constexpr `chrono::minutes` `operator""min` ()
- constexpr `chrono::duration`< long double, `ratio`< 60, 1 > > `operator""min` (long double \_\_mins)
- template<char... \_Digits>  
constexpr `chrono::milliseconds` `operator""ms` ()
- constexpr `chrono::duration`< long double, milli > `operator""ms` (long double \_\_msecs)
- template<char... \_Digits>  
constexpr `chrono::nanoseconds` `operator""ns` ()
- constexpr `chrono::duration`< long double, nano > `operator""ns` (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr `chrono::seconds` `operator""s` ()
- constexpr `chrono::duration`< long double > `operator""s` (long double \_\_secs)
- template<char... \_Digits>  
constexpr `chrono::microseconds` `operator""us` ()
- constexpr `chrono::duration`< long double, micro > `operator""us` (long double \_\_usecs)
- constexpr `chrono::year` `operator""y` (unsigned long long \_\_y) noexcept

#### 4.19.1 Detailed Description

ISO C++ 2014 namespace for suffixes for duration literals.

These suffixes can be used to create `chrono::duration` values with tick periods of hours, minutes, seconds, milliseconds, microseconds or nanoseconds. For example, `std::chrono::seconds(5)` can be written as `5s` after making the suffix visible in the current scope. The suffixes can be made visible by a using-directive or using-declaration such as:

- `using namespace std::chrono_literals;`
- `using namespace std::literals;`
- `using namespace std::chrono;`
- `using namespace std;`
- `using std::chrono_literals::operator""s;`

The result of these suffixes on an integer literal is one of the standard typedefs such as `std::chrono::hours`. The result on a floating-point literal is a duration type with the specified tick period and an unspecified floating-point representation, for example `1.5e2ms` might be equivalent to `chrono::duration<long double, chrono::↵ milli>(1.5e2)`.

Since

C+14



## 4.20 std::placeholders Namespace Reference

### Variables

- const [\\_Placeholder](#)< 1 > [\\_1](#)
- const [\\_Placeholder](#)< 10 > [\\_10](#)
- const [\\_Placeholder](#)< 11 > [\\_11](#)
- const [\\_Placeholder](#)< 12 > [\\_12](#)
- const [\\_Placeholder](#)< 13 > [\\_13](#)
- const [\\_Placeholder](#)< 14 > [\\_14](#)
- const [\\_Placeholder](#)< 15 > [\\_15](#)
- const [\\_Placeholder](#)< 16 > [\\_16](#)
- const [\\_Placeholder](#)< 17 > [\\_17](#)
- const [\\_Placeholder](#)< 18 > [\\_18](#)
- const [\\_Placeholder](#)< 19 > [\\_19](#)
- const [\\_Placeholder](#)< 2 > [\\_2](#)
- const [\\_Placeholder](#)< 20 > [\\_20](#)
- const [\\_Placeholder](#)< 21 > [\\_21](#)
- const [\\_Placeholder](#)< 22 > [\\_22](#)
- const [\\_Placeholder](#)< 23 > [\\_23](#)
- const [\\_Placeholder](#)< 24 > [\\_24](#)
- const [\\_Placeholder](#)< 25 > [\\_25](#)
- const [\\_Placeholder](#)< 26 > [\\_26](#)
- const [\\_Placeholder](#)< 27 > [\\_27](#)
- const [\\_Placeholder](#)< 28 > [\\_28](#)
- const [\\_Placeholder](#)< 29 > [\\_29](#)
- const [\\_Placeholder](#)< 3 > [\\_3](#)
- const [\\_Placeholder](#)< 4 > [\\_4](#)
- const [\\_Placeholder](#)< 5 > [\\_5](#)
- const [\\_Placeholder](#)< 6 > [\\_6](#)
- const [\\_Placeholder](#)< 7 > [\\_7](#)
- const [\\_Placeholder](#)< 8 > [\\_8](#)
- const [\\_Placeholder](#)< 9 > [\\_9](#)

### 4.20.1 Detailed Description

ISO C++ 2011 namespace for std::bind placeholders.

Since

C++11

## 4.21 std::regex\_constants Namespace Reference

### 5.1 Regular Expression Syntax Options

- enum [syntax\\_option\\_type](#) : unsigned int {  
    [\\_S\\_ica](#)se , [\\_S\\_nosubs](#) , [\\_S\\_optimize](#) , [\\_S\\_colla](#)te ,  
    [\\_S\\_ECMAScript](#) , [\\_S\\_ba](#)sic , [\\_S\\_exte](#)nded , [\\_S\\_awk](#) ,  
    [\\_S\\_grep](#) , [\\_S\\_egrep](#) , [\\_S\\_polynomial](#) , [\\_S\\_multiline](#) }
- constexpr [syntax\\_option\\_type](#) [ica](#)se
- constexpr [syntax\\_option\\_type](#) [nosubs](#)
- constexpr [syntax\\_option\\_type](#) [optimize](#)
- constexpr [syntax\\_option\\_type](#) [collate](#)

- constexpr `syntax_option_type` ECMAScript
- constexpr `syntax_option_type` basic
- constexpr `syntax_option_type` extended
- constexpr `syntax_option_type` awk
- constexpr `syntax_option_type` grep
- constexpr `syntax_option_type` egrep
- constexpr `syntax_option_type` multiline
- constexpr `syntax_option_type` \_\_multiline
- constexpr `syntax_option_type` \_\_polynomial
- constexpr `syntax_option_type` operator& (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` operator| (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` operator^ (`syntax_option_type` \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` operator~ (`syntax_option_type` \_\_a) noexcept
- constexpr `syntax_option_type` & operator&= (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` & operator|= (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept
- constexpr `syntax_option_type` & operator^= (`syntax_option_type` & \_\_a, `syntax_option_type` \_\_b) noexcept

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `match_flag_type` : unsigned int {  
`_S_default` , `_S_not_bol` , `_S_not_eol` , `_S_not_bow` ,  
`_S_not_eow` , `_S_any` , `_S_not_null` , `_S_continuous` ,  
`_S_prev_avail` , `_S_sed` , `_S_no_copy` , `_S_first_only` ,  
`_S_match_flag_last` }
- constexpr `match_flag_type` match\_default
- constexpr `match_flag_type` match\_not\_bol
- constexpr `match_flag_type` match\_not\_eol
- constexpr `match_flag_type` match\_not\_bow
- constexpr `match_flag_type` match\_not\_eow
- constexpr `match_flag_type` match\_any
- constexpr `match_flag_type` match\_not\_null
- constexpr `match_flag_type` match\_continuous
- constexpr `match_flag_type` match\_prev\_avail
- constexpr `match_flag_type` format\_default
- constexpr `match_flag_type` format\_sed
- constexpr `match_flag_type` format\_no\_copy
- constexpr `match_flag_type` format\_first\_only
- constexpr `match_flag_type` operator& (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` operator| (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` operator^ (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` operator~ (`match_flag_type` \_\_a) noexcept
- constexpr `match_flag_type` & operator&= (`match_flag_type` & \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & operator|= (`match_flag_type` & \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & operator^= (`match_flag_type` & \_\_a, `match_flag_type` \_\_b) noexcept

### 5.3 Error Types

- enum `error_type` {  
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,  
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,  
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_complexity`,  
`_S_error_stack`, `_S_null`, `_S_grammar` }
- constexpr `error_type error_collate` (`_S_error_collate`)
- constexpr `error_type error_ctype` (`_S_error_ctype`)
- constexpr `error_type error_escape` (`_S_error_escape`)
- constexpr `error_type error_backref` (`_S_error_backref`)
- constexpr `error_type error_brack` (`_S_error_brack`)
- constexpr `error_type error_paren` (`_S_error_paren`)
- constexpr `error_type error_brace` (`_S_error_brace`)
- constexpr `error_type error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type error_range` (`_S_error_range`)
- constexpr `error_type error_space` (`_S_error_space`)
- constexpr `error_type error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type error_complexity` (`_S_error_complexity`)
- constexpr `error_type error_stack` (`_S_error_stack`)

#### 4.21.1 Detailed Description

ISO C++ 2011 namespace for options and flags used with `std::regex`.

#### 4.21.2 Enumeration Type Documentation

##### `error_type`

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

##### `match_flag_type`

```
enum std::regex_constants::match_flag_type : unsigned int
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

##### `syntax_option_type`

```
enum std::regex_constants::syntax_option_type : unsigned int
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### 4.21.3 Function Documentation

##### `error_backref()`

```
error_type std::regex_constants::error_backref (

 _S_error_backref) [inline], [constexpr]
```

The expression contained an invalid back reference.

**error\_badbrace()**

```
error_type std::regex_constants::error_badbrace (
 _S_error_badbrace) [inline], [constexpr]
```

The expression contained an invalid range in a {} expression.

**error\_badrepeat()**

```
error_type std::regex_constants::error_badrepeat (
 _S_error_badrepeat) [inline], [constexpr]
```

One of \*?+{ was not preceded by a valid regular expression.

**error\_brace()**

```
error_type std::regex_constants::error_brace (
 _S_error_brace) [inline], [constexpr]
```

The expression contained mismatched { and }

**error\_brack()**

```
error_type std::regex_constants::error_brack (
 _S_error_brack) [inline], [constexpr]
```

The expression contained mismatched [ and ].

**error\_collate()**

```
error_type std::regex_constants::error_collate (
 _S_error_collate) [inline], [constexpr]
```

The expression contained an invalid collating element name.

**error\_complexity()**

```
error_type std::regex_constants::error_complexity (
 _S_error_complexity) [inline], [constexpr]
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

**error\_ctype()**

```
error_type std::regex_constants::error_ctype (
 _S_error_ctype) [inline], [constexpr]
```

The expression contained an invalid character class name.

**error\_escape()**

```
error_type std::regex_constants::error_escape (
 _S_error_escape) [inline], [constexpr]
```

The expression contained an invalid escaped character, or a trailing escape.

**error\_paren()**

```
error_type std::regex_constants::error_paren (
 _S_error_paren) [inline], [constexpr]
```

The expression contained mismatched ( and ).

**error\_range()**

```
error_type std::regex_constants::error_range (
 _S_error_range) [inline], [constexpr]
```

The expression contained an invalid character range, such as [b-a] in most encodings.

**error\_space()**

```
error_type std::regex_constants::error_space (
 _S_error_space) [inline], [constexpr]
```

There was insufficient memory to convert the expression into a finite state machine.

**error\_stack()**

```
error_type std::regex_constants::error_stack (
 _S_error_stack) [inline], [constexpr]
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

**operator&() [1/2]**

```
match_flag_type std::regex_constants::operator& (
 match_flag_type __a,
 match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator&() [2/2]**

```
syntax_option_type std::regex_constants::operator& (
 syntax_option_type __a,
 syntax_option_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

**operator&=() [1/2]**

```
match_flag_type & std::regex_constants::operator&= (
 match_flag_type & __a,
 match_flag_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

**operator&=() [2/2]**

```
syntax_option_type & std::regex_constants::operator&= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

### **operator^() [1/2]**

```
match_flag_type std::regex_constants::operator^ (
 match_flag_type __a,
 match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

### **operator^() [2/2]**

```
syntax_option_type std::regex_constants::operator^ (
 syntax_option_type __a,
 syntax_option_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

### **operator^=() [1/2]**

```
match_flag_type & std::regex_constants::operator^= (
 match_flag_type & __a,
 match_flag_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

### **operator^=() [2/2]**

```
syntax_option_type & std::regex_constants::operator^= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

### **operator" |() [1/2]**

```
match_flag_type std::regex_constants::operator| (
 match_flag_type __a,
 match_flag_type __b) [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

### **operator" |() [2/2]**

```
syntax_option_type std::regex_constants::operator| (
```

```

syntax_option_type __a,
syntax_option_type __b) [constexpr], [noexcept]

```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### operator" |=( ) [1/2]

```

match_flag_type & std::regex_constants::operator|= (
 match_flag_type & __a,
 match_flag_type __b) [inline], [constexpr], [noexcept]

```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

#### operator" |=( ) [2/2]

```

syntax_option_type & std::regex_constants::operator|= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline], [constexpr], [noexcept]

```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

#### operator~( ) [1/2]

```

match_flag_type std::regex_constants::operator~ (
 match_flag_type __a) [constexpr], [noexcept]

```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

#### operator~( ) [2/2]

```

syntax_option_type std::regex_constants::operator~ (
 syntax_option_type __a) [constexpr], [noexcept]

```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

### 4.21.4 Variable Documentation

#### `__multiline`

```

syntax_option_type std::regex_constants::__multiline [inline], [constexpr]

```

Extension: Equivalent to `regex_constants::multiline` for C++11 and C++14.

#### `__polynomial`

```

syntax_option_type std::regex_constants::__polynomial [inline], [constexpr]

```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception `regex_constants::error_complexity` will be thrown.

### **awk**

`syntax_option_type std::regex_constants::awk [inline], [constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\&apos;`, `\&apos;`, and `\ddd` (where `ddd` is one, two, or three octal digits).

### **basic**

`syntax_option_type std::regex_constants::basic [inline], [constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

### **collate**

`syntax_option_type std::regex_constants::collate [inline], [constexpr]`

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

### **ECMAScript**

`syntax_option_type std::regex_constants::ECMAScript [inline], [constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

### **egrep**

`syntax_option_type std::regex_constants::egrep [inline], [constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that newlines are treated as whitespace.

### **extended**

`syntax_option_type std::regex_constants::extended [inline], [constexpr]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

### **format\_default**

`match_flag_type std::regex_constants::format_default [inline], [constexpr]`

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):



- `$$` The dollar-sign itself (`$`).
- `$&` The matched substring.
- `$`` The portion of the string that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of the string that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is a positive integer less than 100. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::format()`.

### **format\_first\_only**

`match_flag_type` `std::regex_constants::format_first_only` [inline], [constexpr]

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

### **format\_no\_copy**

`match_flag_type` `std::regex_constants::format_no_copy` [inline], [constexpr]

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

### **format\_sed**

`match_flag_type` `std::regex_constants::format_sed` [inline], [constexpr]

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1-2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

### **grep**

`syntax_option_type` `std::regex_constants::grep` [inline], [constexpr]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` `basic`, except that newlines are treated as whitespace.

### **icase**

`syntax_option_type` `std::regex_constants::icase` [inline], [constexpr]

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

### **match\_any**

`match_flag_type` `std::regex_constants::match_any` [inline], [constexpr]

If more than one match is possible then any match is an acceptable result.

### match\_continuous

`match_flag_type` `std::regex_constants::match_continuous` [inline], [constexpr]

The expression only matches a sub-sequence that begins at first .

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

### match\_default

`match_flag_type` `std::regex_constants::match_default` [inline], [constexpr]

The default matching rules.

Referenced by `std::regex_iterator< const char * >::regex_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_token_iterator< const char * >::regex_token_iterator()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_match()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_search()`, `std::regex_search()`, `std::regex_search()`, `std::regex_search()`, `std::regex_search()`, and `std::regex_search()`.

### match\_not\_bol

`match_flag_type` `std::regex_constants::match_not_bol` [inline], [constexpr]

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

### match\_not\_bow

`match_flag_type` `std::regex_constants::match_not_bow` [inline], [constexpr]

The expression \b is not matched against the sub-sequence [first,first).

### match\_not\_eol

`match_flag_type` `std::regex_constants::match_not_eol` [inline], [constexpr]

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

### match\_not\_eow

`match_flag_type` `std::regex_constants::match_not_eow` [inline], [constexpr]

The expression \b should not be matched against the sub-sequence [last,last).

### match\_not\_null

`match_flag_type` `std::regex_constants::match_not_null` [inline], [constexpr]

The expression does not match an empty sequence.

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

### match\_prev\_avail

`match_flag_type` `std::regex_constants::match_prev_avail` [inline], [constexpr]

--first is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the algorithms `regex_match`, `regex_search`, and `regex_replace`, and by the iterators `regex_iterator` and `regex_token_iterator`.

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`.

**multiline**

`syntax_option_type` std::regex\_constants::multiline [inline], [constexpr]

Specifies that the ^ anchor matches at the beginning of a line, and the \$ anchor matches at the end of a line, not only at the beginning/end of the input. Valid for the ECMAScript syntax, ignored otherwise.

Since

C++17

**nosubs**

`syntax_option_type` std::regex\_constants::nosubs [inline], [constexpr]

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied match\_results structure.

**optimize**

`syntax_option_type` std::regex\_constants::optimize [inline], [constexpr]

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

**4.22 std::rel\_ops Namespace Reference****Functions**

- template<class \_Tp>  
bool **operator!=** (const \_Tp &\_\_x, const \_Tp &\_\_y)
- template<class \_Tp>  
bool **operator<=** (const \_Tp &\_\_x, const \_Tp &\_\_y)
- template<class \_Tp>  
bool **operator>** (const \_Tp &\_\_x, const \_Tp &\_\_y)
- template<class \_Tp>  
bool **operator>=** (const \_Tp &\_\_x, const \_Tp &\_\_y)

**4.22.1 Detailed Description**

The generated relational operators are sequestered here.

Libstdc++ headers must not use the contents of `rel_ops`. User code should also avoid them, because unconstrained function templates are too greedy and can easily cause ambiguities.

C++20 default comparisons are a better solution.

**4.22.2 Function Documentation****operator"!=()**

```
template<class _Tp>
bool std::rel_ops::operator!= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines != for arbitrary types, in terms of ==.

**Parameters**

|                          |                |
|--------------------------|----------------|
| $\leftrightarrow$<br>__x | A thing.       |
| $\leftrightarrow$<br>__y | Another thing. |

**Returns**

\_\_x != \_\_y

This function uses == to determine its result.

**operator<=()**

```
template<class _Tp>
bool std::rel_ops::operator<= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines <= for arbitrary types, in terms of <.

**Parameters**

|                          |                |
|--------------------------|----------------|
| $\leftrightarrow$<br>__x | A thing.       |
| $\leftrightarrow$<br>__y | Another thing. |

**Returns**

\_\_x <= \_\_y

This function uses < to determine its result.

**operator>()**

```
template<class _Tp>
bool std::rel_ops::operator> (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines > for arbitrary types, in terms of <.

**Parameters**

|                          |                |
|--------------------------|----------------|
| $\leftrightarrow$<br>__x | A thing.       |
| $\leftrightarrow$<br>__y | Another thing. |

**Returns**

\_\_x > \_\_y

This function uses < to determine its result.

**operator>=()**

```
template<class _Tp>
bool std::rel_ops::operator>= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines >= for arbitrary types, in terms of <.

**Parameters**

|       |                |
|-------|----------------|
| $\_x$ | A thing.       |
| $\_y$ | Another thing. |

**Returns**

$\_x \geq \_y$

This function uses < to determine its result.

**4.23 std::this\_thread Namespace Reference****Functions**

- [thread::id get\\_id \(\)](#) noexcept
- template<typename \_Rep, typename \_Period>  
void [sleep\\_for](#) (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
void [sleep\\_until](#) (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void [yield](#) () noexcept

**4.23.1 Detailed Description**

ISO C++ 2011 namespace for interacting with the current thread.  
C++11 30.3.2 [thread.thread.this] Namespace this\_thread.

**4.23.2 Function Documentation****get\_id()**

```
thread::id std::this_thread::get_id () [inline], [noexcept]
```

The unique identifier of the current thread.

**sleep\_for()**

```
template<typename _Rep, typename _Period>
void std::this_thread::sleep_for (
 const chrono::duration< _Rep, _Period > & __rtime) [inline]
```

this\_thread::sleep\_for

Referenced by [sleep\\_until\(\)](#).

**sleep\_until()**

```
template<typename _Clock, typename _Duration>
void std::this_thread::sleep_until (
 const chrono::time_point< _Clock, _Duration > & __atime) [inline]
this_thread::sleep_until
References sleep_for\(\).
```

**yield()**

```
void std::this_thread::yield () [inline], [noexcept]
Allow the implementation to schedule a different thread.
```

**4.24 std::tr1 Namespace Reference****Namespaces**

- namespace [\\_\\_detail](#)

**Functions**

- template<typename \_Tp>  
[std::complex](#)< \_Tp > **acos** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **acosh** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
\_Tp **arg** (const [complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **asin** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **asinh** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **assoc\_laguerre** (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float **assoc\_laguerref** (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double **assoc\_laguerrel** (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **assoc\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float **assoc\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double **assoc\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **atan** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tp>  
[std::complex](#)< \_Tp > **atanh** (const [std::complex](#)< \_Tp > &)
- template<typename \_Tpx, typename \_Tpy>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpx, \_Tpy >::\_\_type **beta** (\_Tpx \_\_x, \_Tpy \_\_y)
- float **betaf** (float \_\_x, float \_\_y)
- long double **betal** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **comp\_ellint\_1** (\_Tp \_\_k)
- float **comp\_ellint\_1f** (float \_\_k)
- long double **comp\_ellint\_1l** (long double \_\_k)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **comp\_ellint\_2** (\_Tp \_\_k)
- float **comp\_ellint\_2f** (float \_\_k)

- long double **comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **conf\_hyperg** (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float **conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tp>  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type > **conj** (\_Tp \_\_x)
- template<typename \_Tp>  
**std::complex**< \_Tp > **conj** (const **std::complex**< \_Tp > &\_\_z)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **ellint\_1f** (float \_\_k, float \_\_phi)
- long double **ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **ellint\_2f** (float \_\_k, float \_\_phi)
- long double **ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **expint** (\_Tp \_\_x)
- float **expintf** (float \_\_x)
- long double **expintl** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **fabs** (\_Tp \_\_x)
- template<typename \_Tp>  
**std::complex**< \_Tp > **fabs** (const **std::complex**< \_Tp > &\_\_z)
- float **fabs** (float \_\_x)
- long double **fabs** (long double \_\_x)

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergfl (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp>`  
`constexpr _Tp imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)`
- `float legendref (unsigned int __n, float __x)`
- `long double legendrel (unsigned int __n, long double __x)`
- `template<typename _Tp>`  
`_Tp constexpr norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp, typename _Up>`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<typename _Tp>`  
`std::complex< _Tp > pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `float pow (float __x, float __y)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`



- float **sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

#### 4.24.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

#### 4.24.2 Function Documentation

##### arg()

```
template<typename _Tp>
_Tp std::arg (
 const complex< _Tp > & __z) [inline]
```

Return phase angle of z.

##### conf\_hyperg()

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
 _Tpa __a,
 _Tpc __c,
 _Tp __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of real numeratorial parameter a, denominatorial parameter c, and argument x.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

##### Parameters

|                              |                              |
|------------------------------|------------------------------|
| $\longleftrightarrow$<br>__a | The numeratorial parameter   |
| $\longleftrightarrow$<br>__c | The denominatorial parameter |
| $\longleftrightarrow$<br>__x | The argument                 |

**conf\_hypergf()**

```
float __gnu_cxx::conf_hypergf (
 float __a,
 float __c,
 float __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of float numeratorial parameter a, denominatorial parameter c, and argument x.

**See also**

conf\_hyperg for details.

**conf\_hypergl()**

```
long double __gnu_cxx::conf_hypergl (
 long double __a,
 long double __c,
 long double __x) [inline]
```

Return the confluent hypergeometric function  ${}_1F_1(a; c; x)$  of long double numeratorial parameter a, denominatorial parameter c, and argument x.

**See also**

conf\_hyperg for details.

**hyperg()**

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
 _Tpa __a,
 _Tpb __b,
 _Tpc __c,
 _Tp __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of real numeratorial parameters a and b, denominatorial parameter c, and argument x.

The hypergeometric function is defined by

$${}_2F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is  $(x)_k = (x)(x+1)\dots(x+k-1)$ ,  $(x)_0 = 1$

**Parameters**

|                             |                                   |
|-----------------------------|-----------------------------------|
| $\longleftrightarrow$<br>_a | The first numeratorial parameter  |
| $\longleftrightarrow$<br>_b | The second numeratorial parameter |
| $\longleftrightarrow$<br>_c | The denominatorial parameter      |
| $\longleftrightarrow$<br>_x | The argument                      |

**hypergf()**

```
float __gnu_cxx::hypergf (
 float __a,
 float __b,
 float __c,
 float __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**hypergl()**

```
long double __gnu_cxx::hypergl (
 long double __a,
 long double __b,
 long double __c,
 long double __x) [inline]
```

Return the hypergeometric function  ${}_2F_1(a, b; c; x)$  of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

**See also**

hyperg for details.

**norm()**

```
template<typename _Tp>
_Tp constexpr std::norm (
 const complex< _Tp > &) [constexpr]
```

Return z magnitude squared.

**polar()**

```
template<typename _Tp>
complex< _Tp > std::polar (
 const _Tp & __rho,
 const _Tp & __theta = _Tp(0)) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

**4.25 std::tr1::\_\_detail Namespace Reference****4.25.1 Detailed Description**

Implementation details not part of the namespace std::tr1 interface.

**4.26 std::tr2 Namespace Reference****Namespaces**

- namespace [\\_\\_detail](#)

## Classes

- struct [\\_\\_dynamic\\_bitset\\_base](#)
- struct [\\_\\_reflection\\_typelist](#)
- struct [\\_\\_reflection\\_typelist< \\_First, \\_Rest... >](#)
- struct [\\_\\_reflection\\_typelist<>](#)
- struct [bases](#)
- class [bool\\_set](#)
- struct [direct\\_bases](#)
- class [dynamic\\_bitset](#)

## Functions

- bool **certainly** ([bool\\_set](#) \_\_b)
- bool **contains** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- bool **equals** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- bool **is\_emptyset** ([bool\\_set](#) \_\_b)
- bool **is\_indeterminate** ([bool\\_set](#) \_\_b)
- bool **is\_singleton** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator!=** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator!=** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **operator!=** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator&** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_x)
- [bool\\_set](#) **operator==** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator==** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_x)
- [bool\\_set](#) **operator^** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator^** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- bool **possibly** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **set\_complement** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **set\_intersection** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **set\_intersection** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **set\_intersection** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **set\_union** ([bool](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **set\_union** ([bool\\_set](#) \_\_s, [bool](#) \_\_t)
- [bool\\_set](#) **set\_union** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename \_WordT, typename \_Alloc>  
bool **operator!=** (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_lhs, const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_rhs)
- template<typename \_WordT, typename \_Alloc>  
bool **operator<=** (const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_lhs, const [dynamic\\_bitset](#)< \_WordT, \_Alloc > &\_\_rhs)

- `template<typename _WordT, typename _Alloc>`  
`bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

#### 4.26.1 Detailed Description

Namespace for non-standard "TR2" extensions.

## 4.27 std::tr2::\_\_detail Namespace Reference

#### 4.27.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

## 5 Concept Documentation

### 5.1 std::assignable\_from Concept Reference

```
#include <concepts>
```

#### 5.1.1 Concept definition

```
template<typename _Lhs, typename _Rhs>
concept assignable_from = is_lvalue_reference_v<_Lhs>
 && common_reference_with<__detail::__cref<_Lhs>, __detail::__cref<_Rhs>
 && requires(_Lhs __lhs, _Rhs&& __rhs) {
 { __lhs = static_cast<_Rhs&&>(__rhs) } -> same_as<_Lhs>;
 }
```

#### 5.1.2 Detailed Description

[concept.assignable], concept assignable\_from

### 5.2 std::common\_reference\_with Concept Reference

```
#include <concepts>
```

### 5.2.1 Concept definition

```
template<typename _Tp, typename _Up>
concept common_reference_with = same_as<common_reference_t<_Tp, _Up>, common_reference_t<_Up, _Tp>>
 && convertible_to<_Tp, common_reference_t<_Tp, _Up>>
 && convertible_to<_Up, common_reference_t<_Tp, _Up>>
```

### 5.2.2 Detailed Description

[concept.commonref], concept common\_reference\_with

## 5.3 std::common\_with Concept Reference

```
#include <concepts>
```

### 5.3.1 Concept definition

```
template<typename _Tp, typename _Up>
concept common_with = same_as<common_type_t<_Tp, _Up>, common_type_t<_Up, _Tp>>
 && requires {
 static_cast<common_type_t<_Tp, _Up>>(std::declval<_Tp>());
 static_cast<common_type_t<_Tp, _Up>>(std::declval<_Up>());
 }
 && common_reference_with<add_lvalue_reference_t<const _Tp>,
 add_lvalue_reference_t<const _Up>>
 && common_reference_with<add_lvalue_reference_t<common_type_t<_Tp, _Up>>,
 common_reference_t<
 add_lvalue_reference_t<const _Tp>,
 add_lvalue_reference_t<const _Up>>>
```

### 5.3.2 Detailed Description

[concept.common], concept common\_with

## 5.4 std::constructible\_from Concept Reference

```
#include <concepts>
```

### 5.4.1 Concept definition

```
template<typename _Tp, typename... _Args>
concept constructible_from = destructible<_Tp> && is_constructible_v<_Tp, _Args...>
```

### 5.4.2 Detailed Description

[concept.constructible], concept constructible\_from

## 5.5 std::convertible\_to Concept Reference

```
#include <concepts>
```

### 5.5.1 Concept definition

```
template<typename _From, typename _To>
concept convertible_to = is_convertible_v<_From, _To>
 && requires { static_cast<_To>(std::declval<_From>()); } }
```

### 5.5.2 Detailed Description

[concept.convertible], concept convertible\_to

## 5.6 std::copy\_constructible Concept Reference

```
#include <concepts>
```

### 5.6.1 Concept definition

```
template<typename _Tp>
concept copy_constructible = move_constructible<_Tp>
 && constructible_from<_Tp, _Tp&> && convertible_to<_Tp&, _Tp>
 && constructible_from<_Tp, const _Tp&> && convertible_to<const _Tp&, _Tp>
 && constructible_from<_Tp, const _Tp> && convertible_to<const _Tp, _Tp>
```

### 5.6.2 Detailed Description

[concept.copyconstructible], concept copy\_constructible

## 5.7 std::default\_initializable Concept Reference

```
#include <concepts>
```

### 5.7.1 Concept definition

```
template<typename _Tp>
concept default_initializable = constructible_from<_Tp>
 && requires
 {
 _Tp{};
 (void) ::new _Tp;
 }
```

### 5.7.2 Detailed Description

[concept.defaultinitializable], concept default\_initializable

## 5.8 std::derived\_from Concept Reference

```
#include <concepts>
```

### 5.8.1 Concept definition

```
template<typename _Derived, typename _Base>
concept derived_from = __is_base_of(_Base, _Derived)
 && is_convertible_v<const volatile _Derived*, const volatile _Base*>
```

### 5.8.2 Detailed Description

[concept.derived], concept derived\_from

## 5.9 std::destructible Concept Reference

```
#include <concepts>
```

### 5.9.1 Concept definition

```
template<typename _Tp>
concept destructible = __detail::__destructible<_Tp>
```

### 5.9.2 Detailed Description

[concept.destructible], concept destructible

## 5.10 `std::equivalence_relation` Concept Reference

```
#include <concepts>
```

### 5.10.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept equivalence_relation = relation<_Rel, _Tp, _Up>
```

### 5.10.2 Detailed Description

[concept.equiv], concept `equivalence_relation`

## 5.11 `std::indirectly_comparable` Concept Reference

```
#include <iterator_concepts.h>
```

### 5.11.1 Concept definition

```
template<typename _I1, typename _I2, typename _Rel, typename _P1 = identity, typename _P2 = identity>
concept indirectly_comparable = indirect_binary_predicate<_Rel, projected<_I1, _P1>, projected<_I2, _P2>>
```

### 5.11.2 Detailed Description

[alg.req.ind.cmp], concept `indirectly_comparable`

## 5.12 `std::indirectly_copyable` Concept Reference

```
#include <iterator_concepts.h>
```

### 5.12.1 Concept definition

```
template<typename _In, typename _Out>
concept indirectly_copyable = indirectly_readable<_In>
 && indirectly_writable<_Out, iter_reference_t<_In>>
```

### 5.12.2 Detailed Description

[alg.req.ind.copy], concept `indirectly_copyable`

## 5.13 `std::indirectly_movable` Concept Reference

```
#include <iterator_concepts.h>
```

### 5.13.1 Concept definition

```
template<typename _In, typename _Out>
concept indirectly_movable = indirectly_readable<_In>
 && indirectly_writable<_Out, iter_rvalue_reference_t<_In>>
```

### 5.13.2 Detailed Description

[alg.req.ind.move], concept `indirectly_movable`

## 5.14 `std::indirectly_readable` Concept Reference

```
#include <iterator_concepts.h>
```



### 5.14.1 Concept definition

```
template<typename _In>
concept indirectly_readable = __detail::__indirectly_readable_impl<remove_cvref_t<_In>
```

### 5.14.2 Detailed Description

Requirements for types that are readable by applying operator\*.

## 5.15 std::indirectly\_swappable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.15.1 Concept definition

```
template<typename _I1, typename _I2 = _I1>
concept indirectly_swappable = indirectly_readable<_I1> && indirectly_readable<_I2>
 && requires(const _I1 __i1, const _I2 __i2)
{
 ranges::iter_swap(__i1, __i1);
 ranges::iter_swap(__i2, __i2);
 ranges::iter_swap(__i1, __i2);
 ranges::iter_swap(__i2, __i1);
}
```

### 5.15.2 Detailed Description

[alg.req.ind.swap], concept indirectly\_swappable

## 5.16 std::indirectly\_writable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.16.1 Concept definition

```
template<typename _Out, typename _Tp>
concept indirectly_writable = requires(_Out&& __o, _Tp&& __t)
{
 *__o = std::forward<_Tp>(__t);
 *std::forward<_Out>(__o) = std::forward<_Tp>(__t);
 const_cast<const iter_reference_t<_Out>&&>(*__o)
 = std::forward<_Tp>(__t);
 const_cast<const iter_reference_t<_Out>&&>(*std::forward<_Out>(__o))
 = std::forward<_Tp>(__t);
}
```

### 5.16.2 Detailed Description

Requirements for writing a value into an iterator's referenced object.

## 5.17 std::invocable Concept Reference

```
#include <concepts>
```

### 5.17.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept invocable = is_invocable_v<_Fn, _Args...>
```

### 5.17.2 Detailed Description

[concept.invocable], concept invocable

## 5.18 std::mergeable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.18.1 Concept definition

```
template<typename _I1, typename _I2, typename _Out, typename _Rel = ranges::less, typename _P1 = identity,
 typename _P2 = identity>
concept mergeable = input_iterator<_I1> && input_iterator<_I2>
 && weakly_incrementable<_Out> && indirectly_copyable<_I1, _Out>
 && indirectly_copyable<_I2, _Out>
 && indirect_strict_weak_order<_Rel, projected<_I1, _P1>,
 projected<_I2, _P2>>
```

### 5.18.2 Detailed Description

[alg.req.mergeable], concept mergeable

## 5.19 std::move\_constructible Concept Reference

```
#include <concepts>
```

### 5.19.1 Concept definition

```
template<typename _Tp>
concept move_constructible = constructible_from<_Tp, _Tp> && convertible_to<_Tp, _Tp>
```

### 5.19.2 Detailed Description

[concept.moveconstructible], concept move\_constructible

## 5.20 std::permutable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.20.1 Concept definition

```
template<typename _Iter>
concept permutable = forward_iterator<_Iter>
 && indirectly_movable_storable<_Iter, _Iter>
 && indirectly_swappable<_Iter, _Iter>
```

### 5.20.2 Detailed Description

[alg.req.permutable], concept permutable

## 5.21 std::predicate Concept Reference

```
#include <concepts>
```

### 5.21.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept predicate = regular_invocable<_Fn, _Args...>
 && __detail::__boolean_testable<invoke_result_t<_Fn, _Args...>
```

### 5.21.2 Detailed Description

[concept.predicate], concept predicate

## 5.22 std::ranges::bidirectional\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.22.1 Concept definition

```
template<typename _Tp>
concept bidirectional_range = forward_range<_Tp> && bidirectional_iterator<iterator_t<_Tp>
```

### 5.22.2 Detailed Description

A range for which ranges::begin returns a bidirectional iterator.

## 5.23 std::ranges::borrowed\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.23.1 Concept definition

```
template<typename _Tp>
concept borrowed_range = range<_Tp> && __detail::__maybe_borrowed_range<_Tp>
```

### 5.23.2 Detailed Description

[range.range] The borrowed\_range concept.

## 5.24 std::ranges::common\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.24.1 Concept definition

```
template<typename _Tp>
concept common_range = range<_Tp> && same_as<iterator_t<_Tp>, sentinel_t<_Tp>
```

### 5.24.2 Detailed Description

A range for which ranges::begin and ranges::end return the same type.

## 5.25 std::ranges::contiguous\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.25.1 Concept definition

```
template<typename _Tp>
concept contiguous_range = random_access_range<_Tp> && contiguous_iterator<iterator_t<_Tp>
 && requires(_Tp& __t)
{
 { ranges::data(__t) } -> same_as<add_pointer_t<range_reference_t<_Tp>>;
}
```

### 5.25.2 Detailed Description

A range for which ranges::begin returns a contiguous iterator.

## 5.26 std::ranges::forward\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.26.1 Concept definition

```
template<typename _Tp>
concept forward_range = input_range<_Tp> && forward_iterator<iterator_t<_Tp>>
```

### 5.26.2 Detailed Description

A range for which `ranges::begin` returns a forward iterator.

## 5.27 `std::ranges::input_range` Concept Reference

```
#include <ranges_base.h>
```

### 5.27.1 Concept definition

```
template<typename _Tp>
concept input_range = range<_Tp> && input_iterator<iterator_t<_Tp>>
```

### 5.27.2 Detailed Description

A range for which `ranges::begin` returns an input iterator.

## 5.28 `std::ranges::output_range` Concept Reference

```
#include <ranges_base.h>
```

### 5.28.1 Concept definition

```
template<typename _Range, typename _Tp>
concept output_range = range<_Range> && output_iterator<iterator_t<_Range>, _Tp>
```

### 5.28.2 Detailed Description

A range for which `ranges::begin` returns an output iterator.

## 5.29 `std::ranges::random_access_range` Concept Reference

```
#include <ranges_base.h>
```

### 5.29.1 Concept definition

```
template<typename _Tp>
concept random_access_range = bidirectional_range<_Tp> && random_access_iterator<iterator_t<_Tp>>
```

### 5.29.2 Detailed Description

A range for which `ranges::begin` returns a random access iterator.

## 5.30 `std::ranges::range` Concept Reference

```
#include <ranges_base.h>
```

### 5.30.1 Concept definition

```
template<typename _Tp>
concept range = requires(_Tp& __t)
{
 ranges::begin(__t);
 ranges::end(__t);
}
```

### 5.30.2 Detailed Description

[range.range] The range concept.

## 5.31 std::ranges::sized\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.31.1 Concept definition

```
template<typename _Tp>
concept sized_range = range<_Tp>
 && requires(_Tp& __t) { ranges::size(__t); }
```

### 5.31.2 Detailed Description

[range.sized] The sized\_range concept.

## 5.32 std::ranges::view Concept Reference

```
#include <ranges_base.h>
```

### 5.32.1 Concept definition

```
template<typename _Tp>
concept view = range<_Tp> && movable<_Tp> && enable_view<_Tp>
```

### 5.32.2 Detailed Description

[range.view] The ranges::view concept.

## 5.33 std::ranges::viewable\_range Concept Reference

```
#include <ranges_base.h>
```

### 5.33.1 Concept definition

```
template<typename _Tp>
concept viewable_range = range<_Tp>
 && ((view<remove_cvref_t<_Tp>> && constructible_from<remove_cvref_t<_Tp>, _Tp>)
 || (!view<remove_cvref_t<_Tp>>
 && (is_lvalue_reference_v<_Tp>
 || (movable<remove_reference_t<_Tp>>
 && !__detail::__is_initializer_list<remove_cvref_t<_Tp>>))))
```

### 5.33.2 Detailed Description

A range which can be safely converted to a view.

## 5.34 std::regular\_invocable Concept Reference

```
#include <concepts>
```

### 5.34.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept regular_invocable = invocable<_Fn, _Args...>
```

### 5.34.2 Detailed Description

[concept.regularinvocable], concept regular\_invocable

## 5.35 std::relation Concept Reference

```
#include <concepts>
```

### 5.35.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept relation = predicate<_Rel, _Tp, _Tp> && predicate<_Rel, _Up, _Up>
&& predicate<_Rel, _Tp, _Up> && predicate<_Rel, _Up, _Tp>
```

### 5.35.2 Detailed Description

[concept.relation], concept relation

## 5.36 std::same\_as Concept Reference

```
#include <concepts>
```

### 5.36.1 Concept definition

```
template<typename _Tp, typename _Up>
concept same_as = __detail::__same_as<_Tp, _Up> && __detail::__same_as<_Up, _Tp>
```

### 5.36.2 Detailed Description

[concept.same], concept same\_as

## 5.37 std::simd::\_\_vec\_builtin Concept Reference

```
#include <vec_ops.h>
```

### 5.37.1 Concept definition

```
template<typename _Tp>
concept __vec_builtin = __vec_builtin_of<_Tp, remove_cvref_t<decltype(declval<const _Tp>())[0])>>
```

### 5.37.2 Detailed Description

Constrain to any vector builtin.

## 5.38 std::simd::\_\_vec\_builtin\_of Concept Reference

```
#include <vec_ops.h>
```

### 5.38.1 Concept definition

```
template<typename _Tp, typename _ValueType, __simd_size_type _Width = sizeof(_Tp) / sizeof(_ValueType)>
concept __vec_builtin_of = !is_class_v<_Tp> && !is_pointer_v<_Tp> && !is_arithmetic_v<_Tp>
 && __vectorizable<_ValueType>
 && _Width >= 1 && sizeof(_Tp) / sizeof(_ValueType) == _Width
 && same_as<__vec_builtin_type_bytes<_ValueType, sizeof(_Tp)>, _Tp>
 && requires(_Tp& __v, _ValueType __x) { __v[0] = __x; }
```

### 5.38.2 Detailed Description

Constrain to any vector builtin with given value type and optional width.

## 5.39 std::sortable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.39.1 Concept definition

```
template<typename _Iter, typename _Rel = ranges::less, typename _Proj = identity>
concept sortable = permutable<_Iter>
 && indirect_strict_weak_order<_Rel, projected<_Iter, _Proj>
```

### 5.39.2 Detailed Description

[alg.req.sortable], concept sortable

## 5.40 std::strict\_weak\_order Concept Reference

```
#include <concepts>
```

### 5.40.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept strict_weak_order = relation<_Rel, _Tp, _Up>
```

### 5.40.2 Detailed Description

[concept.strictweakorder], concept strict\_weak\_order

## 5.41 std::uniform\_random\_bit\_generator Concept Reference

```
#include <random>
```

### 5.41.1 Concept definition

```
template<typename _Gen>
concept uniform_random_bit_generator = invocable<_Gen&> && unsigned_integral<invoke_result_t<_Gen&>
 && requires
 {
 { _Gen::min() } -> same_as<invoke_result_t<_Gen&>>;
 { _Gen::max() } -> same_as<invoke_result_t<_Gen&>>;
 requires bool_constant<(_Gen::min() < _Gen::max())>::value;
 }
```

### 5.41.2 Detailed Description

Requirements for a uniform random bit generator.

Since

C++20

## 5.42 std::weakly\_incrementable Concept Reference

```
#include <iterator_concepts.h>
```

### 5.42.1 Concept definition

```
template<typename _Iter>
concept weakly_incrementable = movable<_Iter>
 && requires(_Iter __i)
 {
 typename iter_difference_t<_Iter>;
 requires __detail::__is_signed_integer_like<iter_difference_t<_Iter>;
 { ++__i } -> same_as<_Iter&>;
 __i++;
 }
```

### 5.42.2 Detailed Description

Requirements on types that can be incremented with ++.

## 6 Class Documentation

### 6.1 \_\_gnu\_parallel::\_\_accumulate\_binop\_reduct<\_BinOp > Struct Template Reference

```
#include <for_each_selectors.h>
```

#### Public Member Functions

- **\_\_accumulate\_binop\_reduct** (\_BinOp &\_\_b)
- template<typename \_Result, typename \_Addend>  
\_Result **operator()** (const \_Result &\_\_x, const \_Addend &\_\_y)

#### Public Attributes

- \_BinOp &\_\_binop

#### 6.1.1 Detailed Description

```
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct<_BinOp >
```

General reduction, using a binary operator.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 6.2 \_\_gnu\_parallel::\_\_accumulate\_selector<\_It > Struct Template Reference

```
#include <for_each_selectors.h>
```



Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



### Public Member Functions

- `template<typename _Op>`  
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 6.2.1 Detailed Description

`template<typename _It>`  
`struct __gnu_parallel::__accumulate_selector<_It>`

`std::accumulate()` selector.

#### 6.2.2 Member Function Documentation

##### `operator()()`

```

template<typename _It>
template<typename _Op>
std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator()
(
 _Op __o,
 _It __i) [inline]

```

Functor execution.

### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__o</code> | Operator (unused). |
| <code>__i</code> |                    |

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | iterator referencing object. |
| <code>_i</code> |                              |

#### Returns

The current value.

### 6.2.3 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 6.3 `__gnu_parallel::__adjacent_difference_selector< _It >` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector< _It >`:



#### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

#### Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 6.3.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__adjacent_difference_selector< _It >
```

Selector that returns the difference between two adjacent \_\_elements.

### 6.3.2 Member Data Documentation

#### M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:
```

- [for\\_each\\_selectors.h](#)

## 6.4 \_\_gnu\_parallel::\_\_adjacent\_find\_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_adjacent\_find\_selector:



### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

### 6.4.1 Detailed Description

Test predicate on two adjacent elements.

## 6.4.2 Member Function Documentation

### `_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm
(
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

### `operator()()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__adjacent_find_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 6.5 `__gnu_cxx::__alloc_traits<_Alloc, typename >` Struct Template Reference

```
#include <alloc_traits.h>
```

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc, typename >`:



### Public Types

- typedef `std::allocator_traits<_Alloc >_Base_type`
- typedef `_Alloc allocator_type`
- typedef `_Base_type::const_pointer const_pointer`
- typedef `const value_type & const_reference`
- using `const_void_pointer`
- typedef `_Base_type::difference_type difference_type`
- using `is_always_equal`
- typedef `_Base_type::pointer pointer`
- using `propagate_on_container_copy_assignment`
- using `propagate_on_container_move_assignment`
- using `propagate_on_container_swap`
- template<typename `_Tp`>  
using `rebind_alloc`
- template<typename `_Tp`>  
using `rebind_traits`
- typedef `value_type & reference`
- typedef `_Base_type::size_type size_type`
- typedef `_Base_type::value_type value_type`
- using `void_pointer`

### Static Public Member Functions

- static constexpr bool `_S_always_equal ()`
- static constexpr bool `_S_nothrow_move ()`
- static constexpr void `_S_on_swap (_Alloc &__a, _Alloc &__b)`
- static constexpr bool `_S_propagate_on_copy_assign ()`
- static constexpr bool `_S_propagate_on_move_assign ()`
- static constexpr bool `_S_propagate_on_swap ()`
- static constexpr `_Alloc` `_S_select_on_copy (const _Alloc &__a)`
- static constexpr pointer `allocate (_Alloc &__a, size_type __n)`
- static constexpr pointer `allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)`

- static constexpr pointer `allocate` (`_Alloc &__a`, `size_type __n`)
- static constexpr pointer `allocate` (`_Alloc &__a`, `size_type __n`, `const_void_pointer __hint`)
- static constexpr `allocation_result< pointer, size_type >` `allocate_at_least` (`_Alloc &__a`, `size_type __n`)
- template<typename `_Ptr`, typename... `_Args`>  
static constexpr `std::enable_if_t< __is_custom_pointer< _Ptr >::value >` `construct` (`_Alloc &__a`, `_Ptr __p`, `_Args &&... __args`) `noexcept(noexcept( Base_type::construct(__a, std::__to_address(__p), std::forward<__Args>(__args)...))`)
- template<typename `_Tp`, typename... `_Args`>  
static constexpr void `construct` (`_Alloc &__a`, `_Tp *__p`, `_Args &&... __args`) `noexcept( S_nothrow_construct<_Tp, _Args...>())`
- static constexpr void `deallocate` (`_Alloc &__a`, `pointer __p`, `size_type __n`)
- static constexpr void `deallocate` (`_Alloc &__a`, `pointer __p`, `size_type __n`)
- template<typename `_Ptr`>  
static constexpr `std::enable_if_t< __is_custom_pointer< _Ptr >::value >` `destroy` (`_Alloc &__a`, `_Ptr __p`) `noexcept(noexcept( Base_type::destroy(__a, std::__to_address(__p)))`)
- template<typename `_Tp`>  
static constexpr void `destroy` (`_Alloc &__a`, `_Tp *__p`) `noexcept( S_nothrow_destroy<_Tp>())`
- static constexpr `size_type` `max_size` (`const _Alloc &__a`) `noexcept`
- static constexpr `_Alloc` `select_on_container_copy_construction` (`const _Alloc &__rhs`)

### 6.5.1 Detailed Description

```
template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits<_Alloc, typename >
```

Uniform interface to C++98 and C++11 allocators.

### 6.5.2 Member Typedef Documentation

#### `const_void_pointer`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::const_void_pointer [inherited]
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

#### `is_always_equal`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::is_always_equal [inherited]
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

#### `propagate_on_container_copy_assignment`

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::propagate_on_container_copy_assignment [inherited]
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_move\_assignment**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment [inherited]
How the allocator is propagated on move assignment.
Alloc::propagate_on_container_move_assignment if that type exists, otherwise false_type
```

**propagate\_on\_container\_swap**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap [inherited]
How the allocator is propagated on swap.
Alloc::propagate_on_container_swap if that type exists, otherwise false_type
```

**void\_pointer**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer [inherited]
The allocator's void pointer type.
Alloc::void_pointer if that type exists, otherwise pointer_traits<pointer>::rebind<void>
```

**6.5.3 Member Function Documentation****allocate() [1/4]**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr]
Allocate memory.
```

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

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

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
Allocate memory.
```

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

#### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

#### **allocate()** [3/4]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

#### Parameters

|                        |                                              |
|------------------------|----------------------------------------------|
| <code>↔<br/>__a</code> | An allocator.                                |
| <code>↔<br/>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

#### **allocate()** [4/4]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

#### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

#### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`



**allocate\_at\_least()**

```
template<typename _Alloc>
constexpr allocation_result< pointer, size_type > std::allocator_traits< _Alloc >::allocate_at_least (
 _Alloc & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory, generously.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__a</code> | An allocator.                                        |
| <code>__n</code> | The minimum number of objects to allocate space for. |

**Returns**

Memory of suitable size and alignment for `n` or more contiguous objects of type `value_type`.

Returns `a.allocate_at_least(n)` if that expression is well-formed, else `{ a.allocate(n), n }`. When an allocator is obliged to reserve more space than required for the cited `n` objects, it may deliver the extra space to the caller.

**construct()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp, typename... _Args>
constexpr void std::allocator_traits< _Alloc >::construct (
 _Alloc & __a,
 _Tp * __p,
 _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`.

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

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

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr void std::allocator_traits< _Alloc >::deallocate (
 _Alloc & __a,
 pointer __p,
 size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| $\leftrightarrow$<br>_a | An allocator.                                  |
| $\leftrightarrow$<br>_p | Pointer to the memory to deallocate.           |
| $\leftrightarrow$<br>_n | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**deallocate()** [2/2]

```
template<typename _Alloc>
constexpr void std::allocator_traits< _Alloc >::deallocate (
 _Alloc & __a,
 pointer __p,
 size_type __n) [inline], [static], [constexpr], [inherited]
```

Deallocate memory.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| $\leftrightarrow$<br>_a | An allocator.                                  |
| $\leftrightarrow$<br>_p | Pointer to the memory to deallocate.           |
| $\leftrightarrow$<br>_n | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

**destroy()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp>
constexpr void std::allocator_traits< _Alloc >::destroy (
 _Alloc & __a,
 _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Tp`.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| $\leftrightarrow$<br>_a | An allocator.                    |
| $\leftrightarrow$<br>_p | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

**max\_size()**

```
template<typename _Alloc, typename = typename _Alloc::value_type>
```

```
constexpr size_type std::allocator_traits<_Alloc>::max_size (
 const _Alloc & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

#### Parameters

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

#### Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

#### `select_on_container_copy_construction()`

```
template<typename _Alloc>
constexpr _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (
 const _Alloc & __rhs) [inline], [static], [constexpr], [inherited]
```

Obtain an allocator to use when copying a container.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

#### Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

The documentation for this struct was generated from the following file:

- [ext/alloc\\_traits.h](#)

## 6.6 `std::__basic_future<_Res>` Class Template Reference

```
#include <future>
```

Inheritance diagram for `std::__basic_future< _Res >`:



### Public Member Functions

- `__basic_future` (const `__basic_future` &)=delete
- `__basic_future` & **operator=** (const `__basic_future` &)=delete
- bool **valid** () const noexcept
- void **wait** () const
- template<typename `_Rep`, typename `_Period`>  
`future_status wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel) const
- template<typename `_Clock`, typename `_Duration`>  
`future_status wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs) const

### Protected Types

- typedef `__future_base::Result`< `_Res` > & **\_\_result\_type**
- typedef `shared_ptr`< `_State_base` > **\_\_state\_type**

### Protected Member Functions

- `__basic_future` (const `__state_type` &\_\_state)
- `__basic_future` (const `shared_future`< `_Res` > &) noexcept
- `__basic_future` (`future`< `_Res` > &&) noexcept
- `__basic_future` (`shared_future`< `_Res` > &&) noexcept
- `__result_type` `_M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

#### 6.6.1 Detailed Description

```

template<typename _Res>
class std::__basic_future< _Res >

```

Common implementation for `future` and `shared_future`.

## 6.6.2 Member Function Documentation

### `_M_get_result()`

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by `std::future< _Res >::get()`, and `std::shared_future< _Res >::get()`.

The documentation for this class was generated from the following file:

- [future](#)

## 6.7 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



### Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

### Public Member Functions

- `__binder1st` (`const _Operation &__x, const _FirstArgumentType &__y`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`
- `_ResultType operator()` (`const _SecondArgumentType &__x`)

### Protected Attributes

- `_Operation` `_M_op`
- `_FirstArgumentType` `_M_value`

### 6.7.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename
_ResultType>
```

```
class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder1st`, but giving the argument types explicitly.

### 6.7.2 Member Typedef Documentation

#### argument\_type

```
typedef _SecondArgumentType std::unary_function< _SecondArgumentType, _ResultType >::argument_type
type [inherited]
```

`argument_type` is the type of the argument

#### result\_type

```
typedef _ResultType std::unary_function< _SecondArgumentType, _ResultType >::result_type [inherited]
```

`result_type` is the return type

The documentation for this class was generated from the following file:

- [base.h](#)

## 6.8 \_\_gnu\_parallel::\_\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



### Public Types

- typedef `_FirstArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

## Public Member Functions

- **\_\_binder2nd** (const \_Operation &\_\_x, const \_SecondArgumentType &\_\_y)
- \_ResultType **operator()** (\_FirstArgumentType &\_\_x)
- \_ResultType **operator()** (const \_FirstArgumentType &\_\_x) const

## Protected Attributes

- \_Operation **\_M\_op**
- \_SecondArgumentType **\_M\_value**

### 6.8.1 Detailed Description

template<typename \_Operation, typename \_FirstArgumentType, typename \_SecondArgumentType, typename \_ResultType>

class \_\_gnu\_parallel::\_\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >

Similar to std::binder2nd, but giving the argument types explicitly.

### 6.8.2 Member Typedef Documentation

#### argument\_type

```
typedef _FirstArgumentType std::unary_function< _FirstArgumentType, _ResultType >::argument_type
[inherited]
```

argument\_type is the type of the argument

#### result\_type

```
typedef _ResultType std::unary_function< _FirstArgumentType, _ResultType >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [base.h](#)

## 6.9 std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT > Class Template Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >:



## Public Types

- typedef \_ExternT **extern\_type**
- typedef \_InternT **intern\_type**
- typedef codecvt\_base::result **result**
- typedef \_StateT **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Protected Member Functions

- **\_\_codecvt\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const throw () =0
- virtual int **do\_encoding** () const throw () =0
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const throw () =0
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0

## Static Protected Member Functions

- static **\_\_c\_locale \_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static **\_\_c\_locale \_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static **\_\_c\_locale \_S\_lc\_type\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 6.9.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

### 6.9.2 Member Function Documentation

#### do\_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type * __from_next,
```



```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [protected], [pure virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implemented in `std::codecvt<_InternT, _ExternT, _StateT >`, `std::codecvt<_Elem, char, mbstate_t >`, `std::codecvt<_InternT, _ExternT, std::codecvt<char, char, mbstate_t >`, `std::codecvt<char16_t, char, mbstate_t >`, `std::codecvt<char32_t, char, mbstate_t >`, and `std::codecvt<wchar_t, char, mbstate_t >`.

Referenced by [out\(\)](#).

## in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

References [do\\_out\(\)](#).

**unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
```

```
extern_type * __to_end,
extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 6.10 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference

```
#include <mt_allocator.h>
```

### 6.10.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread >
```

Policy for shared `__pool` objects.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 6.11 `__gnu_parallel::__count_if_selector<_It, _Diff >` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff>`:



### Public Member Functions

- `template<typename _Op>`  
`_Diff operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 6.11.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector<_It, _Diff>
```

`std::count_if()` selector.

#### 6.11.2 Member Function Documentation

##### `operator()()`

```
template<typename _It, typename _Diff>
template<typename _Op>
_Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

### Parameters

|                  |           |
|------------------|-----------|
| <code>__o</code> | Operator. |
|------------------|-----------|

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | iterator referencing object. |
| <code>_i</code> |                              |

**Returns**

1 if count, 0 if does not count.

**6.11.3 Member Data Documentation****`_M_finish_iterator`**

```
template<typename _It>
```

```
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**6.12 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference**

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:

**Public Member Functions**

- `template<typename _ValueType>`  
`_Diff operator\(\) (_ValueType &__v, _It __i)`

**Public Attributes**

- `_It \_M\_finish\_iterator`

### 6.12.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector< _It, _Diff >
```

std::count() selector.

### 6.12.2 Member Function Documentation

#### operator()()

```
template<typename _It, typename _Diff>
template<typename _ValueType>
_Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (
 _ValueType & __v,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                                                                                                       |                              |
|-------------------------------------------------------------------------------------------------------|------------------------------|
| <br><code>__v</code> | Current value.               |
| <br><code>__i</code> | iterator referencing object. |

#### Returns

1 if count, 0 if does not count.

### 6.12.3 Member Data Documentation

#### \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.13 std::\_\_ctype\_abstract\_base<\_CharT> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef `_CharT` **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- char **narrow** (**char\_type** \_\_c, char \_\_default) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_default, char \* \_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **scan\_not** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **tolower** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **tolower** (**char\_type** \_\_c) const
- const **char\_type** \* **toupper** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **toupper** (**char\_type** \_\_c) const
- **char\_type** **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, **char\_type** \* \_\_to) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**

- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- `__ctype_abstract_base` (size\_t \_\_refs=0)
- virtual const `char_type` \* `do_is` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, mask \* \_\_vec) const =0
- virtual bool `do_is` (mask \_\_m, `char_type` \_\_c) const =0
- virtual char `do_narrow` (`char_type` \_\_c, char \_\_dfault) const =0
- virtual const `char_type` \* `do_narrow` (const `char_type` \* \_\_lo, const `char_type` \* \_\_hi, char \_\_dfault, char \* \_\_to) const =0
- virtual const `char_type` \* `do_scan_is` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const =0
- virtual const `char_type` \* `do_scan_not` (mask \_\_m, const `char_type` \* \_\_lo, const `char_type` \* \_\_hi) const =0
- virtual const `char_type` \* `do_tolower` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const =0
- virtual `char_type` `do_tolower` (`char_type` \_\_c) const =0
- virtual const `char_type` \* `do_toupper` (`char_type` \* \_\_lo, const `char_type` \* \_\_hi) const =0
- virtual `char_type` `do_toupper` (`char_type` \_\_c) const =0
- virtual `char_type` `do_widen` (char \_\_c) const =0
- virtual const char \* `do_widen` (const char \* \_\_lo, const char \* \_\_hi, `char_type` \* \_\_to) const =0

### Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` & \_\_cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` & \_\_cloc, const char \* \_\_s, `__c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`__c_locale` & \_\_cloc)
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` \_\_cloc, const char \* \_\_s)

#### 6.13.1 Detailed Description

```
template<typename _CharT>
class std::__ctype_abstract_base< _CharT >
```

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

#### 6.13.2 Member Typedef Documentation

##### `char_type`

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base< _CharT >::char_type
```

Typedef for the template parameter.



### 6.13.3 Member Function Documentation

#### do\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [pure virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implemented in [std::ctype<\\_CharT>](#), and [std::mask<\\_CharT>](#).

#### do\_is() [2/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base<_CharT>::do_is (
 mask __m,
 char_type __c) const [protected], [pure virtual]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

#### Returns

$(M \& \text{__m}) \neq 0$ .

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::is\(\)](#), and [std::mask<\\_CharT>::is\(\)](#).

**do\_narrow()** [1/2]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
 char_type __c,
 char __default) const [protected], [pure virtual]
```

Narrow [char\\_type](#) to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, [\\_\\_default](#) is returned instead.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

**Parameters**

|                           |                                           |
|---------------------------|-------------------------------------------|
| <a href="#">__c</a>       | The <a href="#">char_type</a> to convert. |
| <a href="#">__default</a> | Char to return if conversion fails.       |

**Returns**

The converted char.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::narrow\(\)](#), and [std::mask< \\_CharT >::narrow\(\)](#).

**do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [pure virtual]
```

Narrow [char\\_type](#) array to char.

This virtual function converts each [char\\_type](#) in the range [[\\_\\_lo](#),[\\_\\_hi](#)) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, [\\_\\_default](#) is used instead.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

**Parameters**

|                           |                                   |
|---------------------------|-----------------------------------|
| <a href="#">__lo</a>      | Pointer to start of range.        |
| <a href="#">__hi</a>      | Pointer to end of range.          |
| <a href="#">__default</a> | Char to use if conversion fails.  |
| <a href="#">__to</a>      | Pointer to the destination array. |

## Returns

`__hi`.Implemented in [std::ctype<\\_CharT>](#), and [std::mask<\\_CharT>](#).**do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching char\_type if found, else `__hi`.Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), and [std::mask<\\_CharT>](#).Referenced by [std::mask<\\_CharT>::scan\\_is\(\)](#).**do\_scan\_not()**

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which is(`m`,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |

|                                   |                            |
|-----------------------------------|----------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start of range. |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.   |

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::scan\\_not\(\)](#).

**do\_tolower()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                   |                            |
|-----------------------------------|----------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start of range. |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.   |

**Returns**

`__hi`.

Implemented in [std::ctype< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_tolower()** [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
 char_type __c) const [protected], [pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                  |                                        |
|----------------------------------|----------------------------------------|
| <a href="#"><code>__c</code></a> | The <code>char_type</code> to convert. |
|----------------------------------|----------------------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::tolower\(\)](#), and [std::mask<\\_CharT>::tolower\(\)](#).

**do\_toupper()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base<_CharT>::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                        |                            |
|--------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"> __lo</a> | Pointer to start of range. |
| <a href="#"> __hi</a> | Pointer to end of range.   |

**Returns**

\_\_hi.

Implemented in [std::ctype<\\_CharT>](#), and [std::mask<\\_CharT>](#).

**do\_toupper()** [2/2]


```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_toupper (
 char_type __c) const [protected], [pure virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                         |                           |
|---------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"> __c</a> | The char_type to convert. |
|---------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Implemented in [std::ctype<\\_CharT>](#), [std::ctype<wchar\\_t>](#), and [std::mask<\\_CharT>](#).

Referenced by [std::mask<\\_CharT>::toupper\(\)](#), and [std::mask<\\_CharT>::toupper\(\)](#).

**do\_widen()** [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (
 char __c) const [protected], [pure virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                          |                      |
|--------------------------|----------------------|
| $\leftrightarrow$<br>__c | The char to convert. |
|--------------------------|----------------------|

**Returns**

The converted char\_type

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< char >](#), [std::ctype< wchar\\_t >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::widen\(\)](#), and [std::mask< \\_CharT >::widen\(\)](#).

**do\_widen()** [2/2]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [pure virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                           |                                   |
|---------------------------|-----------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start range.           |
| $\leftrightarrow$<br>__hi | Pointer to end of range.          |
| $\leftrightarrow$<br>__to | Pointer to the destination array. |

**Returns**

\_\_hi.

Implemented in [std::ctype< \\_CharT >](#), and [std::mask< \\_CharT >](#).

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

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
```

```
const char_type * __lo,
const char_type * __hi,
mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

#### is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
 mask __m,
 char_type __c) const [inline]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

#### Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>_↵<br/>__c</code> | The char_type to compare the mask of. |
| <code>_↵<br/>__m</code> | The mask to compare against.          |

#### Returns

$(M \& \_m) \neq 0$ .

Referenced by [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#).

#### narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted char.

Referenced by `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::get()`, and `std::time_put<_CharT, _OutIter>::put()`.

**narrow()** [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline]
```

Narrow array to char array.

This function converts each `char_type` in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.



**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_not()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

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

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                                       |                           |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>__c</code> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

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

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Referenced by `std::time_get<_CharT, _Inlter>::get()`.

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

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
 char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted char\_type.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get()`, `std::money_put<_CharT, _Outlter>::do_put()`, and `std::time_put<_CharT, _Outlter>::do_put()`.

**widen()** [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base<_CharT>::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                          |                                   |
|--------------------------|-----------------------------------|
| $\leftrightarrow$<br>_lo | Pointer to start of range.        |
| $\leftrightarrow$<br>_hi | Pointer to end of range.          |
| $\leftrightarrow$<br>_to | Pointer to the destination array. |

**Returns**

$\leftrightarrow$  \_hi.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**6.14 std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc > Struct Template Reference**

```
#include <dynamic_bitset>
```

Inheritance diagram for std::tr2::\_\_dynamic\_bitset\_base< \_WordT, \_Alloc >:

**Public Types**

- typedef \_Alloc **allocator\_type**
- typedef \_WordT **block\_type**
- typedef size\_t **size\_type**

**Public Member Functions**

- **\_\_dynamic\_bitset\_base** (\_\_dynamic\_bitset\_base && \_\_b)=default
- **\_\_dynamic\_bitset\_base** (const \_\_dynamic\_bitset\_base &)=default
- **\_\_dynamic\_bitset\_base** (const allocator\_type &\_\_alloc)
- **\_\_dynamic\_bitset\_base** (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type & $\leftrightarrow$  alloc=allocator\_type())

- `size_t _M_are_all_aux ()` const noexcept
- `void _M_clear ()` noexcept
- `void _M_do_and (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_append_block (block_type __block, size_type __pos)`
- `size_t _M_do_count ()` const noexcept
- `void _M_do_dif (const __dynamic_bitset_base &__x)` noexcept
- `size_type _M_do_find_first (size_t __not_found)` const
- `size_type _M_do_find_next (size_t __prev, size_t __not_found)` const
- `void _M_do_flip ()` noexcept
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_reset ()` noexcept
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()` noexcept
- `unsigned long long _M_do_to_ullong ()` const
- `unsigned long _M_do_to_ulong ()` const
- `void _M_do_xor (const __dynamic_bitset_base &__x)` noexcept
- `allocator_type _M_get_allocator ()` const noexcept
- `block_type _M_getword (size_type __pos)` const noexcept
- `block_type & _M_getword (size_type __pos)` noexcept
- `block_type _M_hiword ()` const noexcept
- `block_type & _M_hiword ()` noexcept
- `bool _M_is_any ()` const noexcept
- `bool _M_is_equal (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_less (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_proper_subset_of (const __dynamic_bitset_base &__b)` const noexcept
- `bool _M_is_subset_of (const __dynamic_bitset_base &__b)` noexcept
- `void _M_resize (size_t __nbits, bool __value)`
- `size_type _M_size ()` const noexcept
- `void _M_swap (__dynamic_bitset_base &__b)` noexcept
- `__dynamic_bitset_base & operator= (__dynamic_bitset_base &&)=default`
- `__dynamic_bitset_base & operator= (const __dynamic_bitset_base &)=default`

### Static Public Member Functions

- `static block_type _S_maskbit (size_type __pos)` noexcept
- `static size_type _S_whichbit (size_type __pos)` noexcept
- `static size_type _S_whichbyte (size_type __pos)` noexcept
- `static size_type _S_whichword (size_type __pos)` noexcept

### Public Attributes

- `std::vector< block_type, allocator_type > _M_w`

### Static Public Attributes

- `static const size_type _S_bits_per_block`
- `static const size_type npos`

### 6.14.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
```

Base class, general case.

See documentation for `dynamic_bitset`.

### 6.14.2 Member Data Documentation

#### `_M_w`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
0 is the least significant word.
```

The documentation for this struct was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

## 6.15 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



### Public Member Functions

- `template<typename _ValueType>`  
`bool operator() (_ValueType &__v, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

### 6.15.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__fill_selector<_It>
```

`std::fill()` selector.

### 6.15.2 Member Function Documentation

#### operator>()

```
template<typename _It>
template<typename _ValueType>
bool __gnu_parallel::__fill_selector<_It>::operator() (
 _ValueType & __v,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                                  |                              |
|----------------------------------|------------------------------|
| <a href="#"><code>__v</code></a> | Current value.               |
| <a href="#"><code>__i</code></a> | iterator referencing object. |

### 6.15.3 Member Data Documentation

#### \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.16 \_\_gnu\_parallel::\_\_find\_first\_of\_selector<\_FIterator> Struct Template Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_find\_first\_of\_selector<\_FIterator>:



## Public Member Functions

- **`__find_first_of_selector`** (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > \_\_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator() ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## Public Attributes

- `_FIterator \_\_M\_begin`
- `_FIterator \_\_M\_end`

### 6.16.1 Detailed Description

```
template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector< _FIterator >
```

Test predicate on several elements.

### 6.16.2 Member Function Documentation

#### [\\_\\_M\\_sequential\\_algorithm\(\)](#)

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_first_of_selector< _FIterator >::__M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

#### [operator>\(\)\(\)](#)

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.



**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

**6.17 \_\_gnu\_parallel::\_\_find\_if\_selector Struct Reference**

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_find\_if\_selector:

**Public Member Functions**

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair<_RAIter1, _RAIter2> \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

**6.17.1 Detailed Description**

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

**6.17.2 Member Function Documentation****`_M_sequential_algorithm()`**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
```

```

 _RAIter2 __begin2,
 _Pred __pred) [inline]

```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

References [std::make\\_pair\(\)](#).

#### `operator>()()`

```

template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_if_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]

```

Test on one position.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 6.18 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_for\_each\_selector<\_It>:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 6.18.1 Detailed Description

`template<typename _It>`  
`struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

#### 6.18.2 Member Function Documentation

##### `operator()()`

```

template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]

```

Functor execution.

### Parameters

|                  |           |
|------------------|-----------|
| <code>__o</code> | Operator. |
| <code>__i</code> |           |

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | iterator referencing object. |
| <code>_i</code> |                              |

### 6.18.3 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 6.19 `__cxxabiv1::__forced_unwind` Class Reference

```
#include <cxxabi_forced.h>
```

#### 6.19.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

### 6.20 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



**Public Member Functions**

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

**Public Attributes**

- `_It _M_finish_iterator`

**6.20.1 Detailed Description**

`template<typename _It>`  
`struct __gnu_parallel::__generate_selector< _It >`

`std::generate()` selector.

**6.20.2 Member Function Documentation****operator>()()**

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector< _It >::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

**6.20.3 Member Data Documentation****\_M\_finish\_iterator**

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

**6.21 \_\_gnu\_parallel::\_\_generic\_find\_selector Struct Reference**

```
#include <find_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



#### 6.21.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

#### 6.22 `__gnu_parallel::__generic_for_each_selector<_It>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>:



#### Public Attributes

- [\\_It\\_M\\_finish\\_iterator](#)

### 6.22.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generic_for_each_selector< _It >
```

Generic \_\_selector for embarrassingly parallel functions.

### 6.22.2 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:
```

- [for\\_each\\_selectors.h](#)

## 6.23 `__gnu_parallel::__identity_selector< _It >` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__identity_selector< _It >`:



### Public Member Functions

- `template<typename _Op>`  
`_It operator() (_Op __o, _It __i)`

### Public Attributes

- `_It` [\\_M\\_finish\\_iterator](#)

### 6.23.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__identity_selector< _It >
```

Selector that just returns the passed iterator.



### 6.23.2 Member Function Documentation

#### operator>()

```
template<typename _It>
template<typename _Op>
_It __gnu_parallel::__identity_selector<_It>::operator() (
 _Op __o,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                                                                                                    |                              |
|----------------------------------------------------------------------------------------------------|------------------------------|
|  <code>__o</code> | Operator (unused).           |
|  <code>__i</code> | iterator referencing object. |

#### Returns

Passed iterator.

### 6.23.3 Member Data Documentation

#### \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.24 \_\_gnu\_parallel::\_\_inner\_product\_selector<\_It, \_It2, \_Tp> Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`:



### Public Member Functions

- `__inner_product_selector` (`_It __b1, _It2 __b2`)
- `template<typename _Op>`  
`_Tp operator()` (`_Op __mult, _It __current`)

### Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

#### 6.24.1 Detailed Description

```
template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>
```

`std::inner_product()` selector.

#### 6.24.2 Constructor & Destructor Documentation

`__inner_product_selector()`

```
template<typename _It, typename _It2, typename _Tp>
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (
 _It __b1,
 _It2 __b2) [inline], [explicit]
```

Constructor.

### Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__b1</code> | Begin iterator of first sequence.  |
| <code>__b2</code> | Begin iterator of second sequence. |

References [\\_\\_begin1\\_iterator](#), and [\\_\\_begin2\\_iterator](#).

### 6.24.3 Member Function Documentation

#### operator>()

```
template<typename _It, typename _It2, typename _Tp>
template<typename _Op>
_Tp __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator() (
 _Op __mult,
 _It __current) [inline]
```

Functor execution.

#### Parameters

|                        |                              |
|------------------------|------------------------------|
| <code>__mult</code>    | Multiplication functor.      |
| <code>__current</code> | iterator referencing object. |

#### Returns

Inner product elemental `__result`.

References [\\_\\_begin1\\_iterator](#), and [\\_\\_begin2\\_iterator](#).

### 6.24.4 Member Data Documentation

#### `__begin1_iterator`

```
template<typename _It, typename _It2, typename _Tp>
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.  
Referenced by [\\_\\_inner\\_product\\_selector\(\)](#), and [operator\(\)](#).

#### `__begin2_iterator`

```
template<typename _It, typename _It2, typename _Tp>
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.  
Referenced by [\\_\\_inner\\_product\\_selector\(\)](#), and [operator\(\)](#).

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).  
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.25 std::\_\_is\_fast\_hash<\_Hash> Struct Template Reference

```
#include <functional_hash.h>
```

### 6.25.1 Detailed Description

```
template<typename _Hash>
struct std::__is_fast_hash< _Hash >
```

Hint about performance of hash functions.

If a given hash function object is not fast, the hash-based containers will cache the hash code. The default behavior is to consider that hashers are fast unless specified otherwise.

Users can specialize this for their own hash functions in order to force caching of hash codes in unordered containers. Specializing this trait affects the ABI of the unordered containers, so use it carefully.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 6.26 std::\_\_is\_location\_invariant< \_Tp > Struct Template Reference

```
#include <std_function.h>
```

### 6.26.1 Detailed Description

```
template<typename _Tp>
struct std::__is_location_invariant< _Tp >
```

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Trivially copyable types are location-invariant and users can specialize this trait for other types.

The documentation for this struct was generated from the following file:

- [std\\_function.h](#)

## 6.27 \_\_gnu\_parallel::\_\_max\_element\_reduct< \_Compare, \_It > Struct Template Reference

```
#include <for_each_selectors.h>
```

### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

### 6.27.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct< _Compare, _It >
```

Reduction for finding the maximum element, using a comparator.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.28 \_\_gnu\_parallel::\_\_min\_element\_reduct< \_Compare, \_It > Struct Template Reference

```
#include <for_each_selectors.h>
```

**Public Member Functions**

- `__min_element_reduct` (`_Compare` & `__c`)
- `_lt operator()` (`_lt __x`, `_lt __y`)

**Public Attributes**

- `_Compare` & `__comp`

**6.28.1 Detailed Description**

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.  
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**6.29 `__gnu_cxx::__detail::__mini_vector<_Tp>` Class Template Reference**

```
#include <bitmap_allocator.h>
```

**Public Types**

- typedef const `_Tp` & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef pointer **iterator**
- typedef `_Tp` \* **pointer**
- typedef `_Tp` & **reference**
- typedef std::size\_t **size\_type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator `__pos`) throw ()
- void **insert** (iterator `__pos`, const\_reference `__x`)
- reference **operator[]** (const size\_type `__pos`) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference `__x`)
- size\_type **size** () const throw ()

**6.29.1 Detailed Description**

```
template<typename _Tp>
class __gnu_cxx::__detail::__mini_vector<_Tp>
```

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.  
It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present.

2. Used ONLY for PODs.
3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

### 6.30 \_\_gnu\_parallel::\_\_mismatch\_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_\_mismatch\_selector:



#### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

#### 6.30.1 Detailed Description

Test inverted predicate on a single element.

#### 6.30.2 Member Function Documentation

##### `_M_sequential_algorithm()`

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

**Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

**operator()()**

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__mismatch_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

**6.31 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference**

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



## Public Types

- typedef `_Poolp` **\_\_policy\_type**
- typedef `_Poolp::pool_type` **\_\_pool\_type**
- typedef `const _Tp *` **const\_pointer**
- typedef `const _Tp &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Tp *` **pointer**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `_Tp &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **\_\_mt\_alloc** (const `__mt_alloc &`) noexcept
- template<typename `_Tp1`, typename `_Poolp1`>  
**\_\_mt\_alloc** (const `__mt_alloc<_Tp1, _Poolp1 > &`) noexcept
- const `__pool_base::_Tune` **M\_get\_options** ()
- void **M\_set\_options** (`__pool_base::_Tune` `_t`)
- const\_pointer **address** (const\_reference `__x`) const noexcept
- pointer **address** (reference `__x`) const noexcept
- pointer **allocate** (size\_type `_n`, const void `*=0`)
- template<typename `_Up`, typename... `_Args`>  
void **construct** (`_Up *` `_p`, `_Args &&... __args`)
- void **deallocate** (pointer `_p`, size\_type `_n`)
- template<typename `_Up`>  
void **destroy** (`_Up *` `_p`)
- size\_type **max\_size** () const noexcept

### 6.31.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt\\_allocator.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html)

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 6.32 \_\_gnu\_cxx::\_\_mt\_alloc\_base<\_Tp> Class Template Reference

```
#include <mt_allocator.h>
```



Inheritance diagram for `__gnu_cxx::__mt_alloc_base< _Tp >`:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename \_Up>  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

#### 6.32.1 Detailed Description

```

template<typename _Tp>
class __gnu_cxx::__mt_alloc_base< _Tp >

```

Base class for `_Tp` dependent member functions.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

### 6.33 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, \_RAIter3 \_\_length, \_Compare \_\_comp)

### 6.33.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
```

```
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with \_\_sentinels turned off.

Note that 3-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.34 \_\_gnu\_parallel::\_\_multiway\_merge\_3\_variant\_sentinel\_switch< true, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare > Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, \_RAIter3 \_\_length, \_Compare \_\_comp)

### 6.34.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
```

```
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with \_\_sentinels turned on.

Note that 3-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.35 \_\_gnu\_parallel::\_\_multiway\_merge\_4\_variant\_sentinel\_switch< \_\_sentinels, \_RAIterIterator, \_RAIter3, \_DifferenceTp, \_Compare > Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- **\_RAIter3 operator()** (\_RAIterIterator \_\_seqs\_begin, \_RAIterIterator \_\_seqs\_end, \_RAIter3 \_\_target, \_RAIter3 \_\_length, \_Compare \_\_comp)

### 6.35.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
```

```
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with \_\_sentinels turned off.

Note that 4-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.36 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _↵ DifferenceTp __length, _Compare __comp)`

#### 6.36.1 Detailed Description

```
template<typename _RAIterliterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterliterator, _RAIter3, _↵
_DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.37 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- `_RAIter3 operator() (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, const type- name std::iterator\_traits< typename std::iterator\_traits< _RAIterliterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

#### 6.37.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterliterator, typename _RAIter3, typename _↵
_DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterliterator,
_RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned on.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.38 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterliterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

```
#include <multiway_merge.h>
```

## Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

### 6.38.1 Detailed Description

`template<bool __stable, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>`

`struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for k-way merging with \_\_sentinels turned off.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.39 std::\_\_new\_allocator< \_Tp > Class Template Reference

```
#include <memory>
```

Inheritance diagram for `std::__new_allocator< _Tp >`:



## Public Types

- `typedef std::ptrdiff_t difference_type`
- `typedef std::true\_type propagate_on_container_move_assignment`

- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- constexpr **\_\_new\_allocator** (const `__new_allocator` &) noexcept
- template<typename `_Tp1`>  
constexpr **\_\_new\_allocator** (const `__new_allocator`< `_Tp1` > &) noexcept
- constexpr `_Tp` \* **allocate** (size\_type `_n`, const void \*=`static_cast`< const void \* >(0))
- constexpr `std::allocation_result`< `_Tp` \*, size\_t > **allocate\_at\_least** (size\_t `_n`)
- constexpr void **deallocate** (`_Tp` \*`_p`, size\_type `_n`)
- `__new_allocator` & **operator=** (const `__new_allocator` &)=default

### Static Public Member Functions

- static constexpr void **\_S\_check\_allocation\_limit** (size\_t `_n`)

### Friends

- template<typename `_Up`>  
constexpr bool **operator==** (const `__new_allocator` &, const `__new_allocator`< `_Up` > &) noexcept

#### 6.39.1 Detailed Description

template<typename `_Tp`>  
class `std::__new_allocator`< `_Tp` >

An allocator that uses global `new`, as per C++03 [20.4.1].  
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is the default base-class implementation of `std::allocator`, and is also the base-class of the `__gnu_cxx::new_allocator` extension. You should use either `std::allocator` or `__gnu_cxx::new_allocator` instead of using this directly.

### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [bits/new\\_allocator.h](#)

## 6.40 `std::__numeric_limits_base` Struct Reference

```
#include <limits>
```

Inheritance diagram for `std::__numeric_limits_base`:



### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)

- static constexpr [float\\_denorm\\_style](#) `has_denorm`
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) `round_style`
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

### 6.40.1 Detailed Description

Part of `std::numeric_limits`.

The `static const` members are usable as integral constant expressions.

#### Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

### 6.40.2 Member Data Documentation

#### `digits`

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Referenced by [std::generate\\_canonical\(\)](#), [std::independent\\_bits\\_engine<\\_RandomNumberEngine, \\_\\_w, \\_UIntType >::operator\(\)\(\)](#), and [std::linear\\_congruential\\_engine<\\_UIntType, \\_\\_a, \\_\\_c, \\_\\_m >::seed\(\)](#).

#### `digits10`

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### `has_denorm`

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See [std::float\\_denorm\\_style](#) for more information.

**has\_denorm\_loss**

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

**has\_infinity**

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

**has\_quiet\_NaN**

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

**has\_signaling\_NaN**

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

**is\_bounded**

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

**is\_exact**

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

**is\_iec559**

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```

True if the type is integer.

**is\_modulo**

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```

True if the type is signed.



**is\_specialized**

```
bool std::__numeric_limits_base::is_specialized [static], [constexpr]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

```
int std::__numeric_limits_base::max_digits10 [static], [constexpr]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

```
int std::__numeric_limits_base::max_exponent [static], [constexpr]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

```
int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

```
int std::__numeric_limits_base::min_exponent [static], [constexpr]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

```
int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

```
int std::__numeric_limits_base::radix [static], [constexpr]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Referenced by [std::generate\\_canonical\(\)](#).

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

## 6.41 `__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread >` Struct Template Reference

```
#include <mt_allocator.h>
```

### 6.41.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread >
```

Policy for individual `__pool` objects.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 6.42 `__gnu_cxx::__pool<_Thread >` Class Template Reference

### 6.42.1 Detailed Description

```
template<bool _Thread>
class __gnu_cxx::__pool<_Thread >
```

Data describing the underlying memory pool, parameterized on threading support.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 6.43 `__gnu_cxx::__pool<false >` Class Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool<false >`:



### Public Types

- typedef unsigned short int **`_Binmap_type`**
- typedef std::size\_t **`size_t`**

**Public Member Functions**

- `__pool` (const `__pool_base::__Tune &__tune`)
- void `_M_adjust_freelist` (const `_Bin_record &`, `_Block_record *`, `size_t`)
- bool `_M_check_threshold` (`size_t __bytes`)
- void `_M_destroy` () throw ()
- `size_t _M_get_align` ()
- const `_Bin_record & _M_get_bin` (`size_t __which`)
- `size_t _M_get_binmap` (`size_t __bytes`)
- const `_Tune & _M_get_options` () const
- `size_t _M_get_thread_id` ()
- void `_M_initialize_once` ()
- void `_M_reclaim_block` (`char *__p`, `size_t __bytes`) throw ()
- `char * _M_reserve_block` (`size_t __bytes`, const `size_t __thread_id`)
- void `_M_set_options` (`_Tune __t`)

**Protected Attributes**

- `_Binmap_type * _M_binmap`
- bool `_M_init`
- `_Tune _M_options`

**6.43.1 Detailed Description**

Specialization for single thread.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

**6.44 `__gnu_cxx::__pool< true >` Class Reference**

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool< true >`:

**Public Types**

- typedef unsigned short int `_Binmap_type`
- typedef std::size\_t `size_t`

## Public Member Functions

- `__pool` (const `__pool_base::__Tune &__tune`)
- void `_M_adjust_freelist` (const `_Bin_record &__bin`, `_Block_record * __block`, `size_t __thread_id`)
- bool `_M_check_threshold` (`size_t __bytes`)
- void `_M_destroy` () throw ()
- void `_M_destroy_thread_key` (`void *`) throw ()
- `size_t _M_get_align` ()
- const `_Bin_record & _M_get_bin` (`size_t __which`)
- `size_t _M_get_binmap` (`size_t __bytes`)
- const `_Tune & _M_get_options` () const
- `size_t _M_get_thread_id` ()
- void `_M_initialize` (`__destroy_handler`)
- void `_M_initialize_once` ()
- void `_M_reclaim_block` (`char * __p`, `size_t __bytes`) throw ()
- `char * _M_reserve_block` (`size_t __bytes`, const `size_t __thread_id`)
- void `_M_set_options` (`_Tune __t`)

## Protected Attributes

- `_Binmap_type * _M_binmap`
- bool `_M_init`
- `_Tune _M_options`

### 6.44.1 Detailed Description

Specialization for thread enabled, via `gthreads.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 6.45 `__gnu_cxx::__pool_alloc<_Tp>` Class Template Reference

```
#include <pool_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool_alloc<_Tp>`:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **\_\_pool\_alloc** (const \_\_pool\_alloc &) noexcept
- template<typename \_Tp1>  
  **\_\_pool\_alloc** (const \_\_pool\_alloc< \_Tp1 > &) noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \* = 0)
- template<typename \_Up, typename... \_Args>  
  void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up>  
  void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept

### 6.45.1 Detailed Description

template<typename \_Tp>  
class \_\_gnu\_cxx::\_\_pool\_alloc< \_Tp >

Allocator using a memory pool with a single lock.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 6.46 \_\_gnu\_cxx::\_\_pool\_alloc\_base Class Reference

```
#include <pool_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc\_base:



### Protected Types

- enum { **\_S\_free\_list\_size** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_align** }

### Protected Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Protected Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

#### 6.46.1 Detailed Description

Base class for \_\_pool\_alloc.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties:

1. If globally mandated, then allocate objects from new
2. If the clients request an object of size > \_S\_max\_bytes, the resulting object will be obtained directly from new
3. In all other cases, we allocate an object of size exactly \_S\_round\_up(requested\_size). Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

#### 6.47 \_\_gnu\_cxx::\_\_pool\_base Struct Reference

```
#include <mt_allocator.h>
```

Inheritance diagram for `__gnu_cxx::__pool_base`:



### Public Types

- typedef unsigned short int **\_Binmap\_type**
- typedef std::size\_t **size\_t**

### Public Member Functions

- **\_\_pool\_base** (const \_Tune &\_\_options)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- size\_t **\_M\_get\_align** ()
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- void **\_M\_set\_options** (\_Tune \_\_t)

### Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

#### 6.47.1 Detailed Description

Base class for pool object.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 6.48 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

```
#include <rc_string_base.h>
```

Inheritance diagram for `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>`:



## Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility<_CharT, _Traits, _Alloc> _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

## Public Member Functions

- `__rc_string_base (__rc_string_base &&__rcs)`
- `template<typename _InputIterator>  
__rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`
- `__rc_string_base (const __rc_string_base &__rcs)`



- `__rc_string_base` (const `_Alloc` & `_a`)
- `__rc_string_base` (size\_type `__n`, `_CharT` `__c`, const `_Alloc` & `_a`)
- void `_M_assign` (const `__rc_string_base` & `_rcs`)
- size\_type `_M_capacity` () const
- void `_M_clear` ()
- bool `_M_compare` (const `__rc_string_base` &) const
- bool `_M_compare` (const `__rc_string_base` & `_rcs`) const
- bool `_M_compare` (const `__rc_string_base` & `_rcs`) const
- `_CharT` \* `_M_data` () const
- void `_M_erase` (size\_type `__pos`, size\_type `__n`)
- allocator\_type & `_M_get_allocator` ()
- const allocator\_type & `_M_get_allocator` () const
- bool `_M_is_shared` () const
- void `_M_leak` ()
- size\_type `_M_length` () const
- size\_type `_M_max_size` () const
- void `_M_mutate` (size\_type `__pos`, size\_type `__len1`, const `_CharT` \* `__s`, size\_type `__len2`)
- void `_M_reserve` (size\_type `__res`)
- void `_M_set_leaked` ()
- void `_M_set_length` (size\_type `__n`)
- void `_M_swap` (`__rc_string_base` & `_rcs`)
- template<typename `_InIterator`>  
`_CharT` \* `_S_construct` (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` & `_a`, [std::forward\\_iterator\\_tag](#))

### Protected Types

- typedef `__gnu_cxx::__normal_iterator< const_pointer, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_\_↵`  
`__rc_string_base>>` **`__const_rc_iterator`**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_\_↵`  
`__sso_string_base>>` **`__const_sso_iterator`**
- typedef `__gnu_cxx::__normal_iterator< pointer, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_\_rc↵`  
`string_base>>` **`__rc_iterator`**
- typedef `__gnu_cxx::__normal_iterator< pointer, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_\_sso↵`  
`string_base>>` **`__sso_iterator`**
- typedef `__alloc_traits<_CharT_alloc_type>` **`_CharT_alloc_traits`**
- typedef `_CharT_alloc_traits::const_pointer` **`const_pointer`**
- typedef `_CharT_alloc_type::difference_type` **`difference_type`**
- typedef `_CharT_alloc_traits::pointer` **`pointer`**

### Static Protected Member Functions

- static void `_S_assign` (`_CharT` \* `__d`, size\_type `__n`, `_CharT` `__c`)
- static int `_S_compare` (size\_type `__n1`, size\_type `__n2`)
- static void `_S_copy` (`_CharT` \* `__d`, const `_CharT` \* `__s`, size\_type `__n`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__const_rc_iterator` `__k1`, `__const_rc_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__const_sso_iterator` `__k1`, `__const_sso_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__rc_iterator` `__k1`, `__rc_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `__sso_iterator` `__k1`, `__sso_iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, `_CharT` \* `__k1`, `_CharT` \* `__k2`)
- template<typename `_Iterator`>  
static void `_S_copy_chars` (`_CharT` \* `__p`, `_Iterator` `__k1`, `_Iterator` `__k2`)
- static void `_S_copy_chars` (`_CharT` \* `__p`, const `_CharT` \* `__k1`, const `_CharT` \* `__k2`)
- static void `_S_move` (`_CharT` \* `__d`, const `_CharT` \* `__s`, size\_type `__n`)

### 6.48.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >
```

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

 [_Rep]
 _M_length
[_rc_string_base<char_type>] _M_capacity
_M_datapointer _M_refcount
_M_p -----> unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 6.49 std::tr2::\_\_reflection\_typelist< \_Elements > Struct Template Reference

### 6.49.1 Detailed Description

```
template<typename... _Elements>
struct std::tr2::__reflection_typelist< _Elements >
```

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 6.50 std::tr2::\_\_reflection\_typelist< \_First, \_Rest... > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- typedef [std::false\\_type](#) `empty`

### 6.50.1 Detailed Description

```
template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist< _First, _Rest... >
```

Partial specialization.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 6.51 `std::tr2::__reflection_typelist<>` Struct Reference

```
#include <type_traits>
```

### Public Types

- typedef [std::true\\_type](#) `empty`

#### 6.51.1 Detailed Description

Specialization for an empty typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 6.52 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



### Public Member Functions

- [\\_\\_replace\\_if\\_selector](#) (const `_Tp` & [\\_\\_new\\_val](#))
- bool [operator\(\)](#) (`_Op` & `__o`, `_It` `__i`)

### Public Attributes

- const `_Tp` & [\\_\\_new\\_val](#)
- `_It` [\\_M\\_finish\\_iterator](#)

#### 6.52.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>
```

`std::replace()` selector.

## 6.52.2 Constructor & Destructor Documentation

### `__replace_if_selector()`

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
 const _Tp & __new_val) [inline], [explicit]
```

Constructor.

#### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

References `__new_val`.

## 6.52.3 Member Function Documentation

### `operator>()()`

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

References `__new_val`.

## 6.52.4 Member Data Documentation

### `__new_val`

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val
```

Value to replace with.

Referenced by `__replace_if_selector()`, and `operator>()()`.

### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

## 6.53 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



### Public Member Functions

- `__replace_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Tp` & `__v`, `_It` `__i`)

### Public Attributes

- const `_Tp` & `__new_val`
- `_It` `_M_finish_iterator`

#### 6.53.1 Detailed Description

```
template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector<_It, _Tp>
```

`std::replace()` selector.

#### 6.53.2 Constructor & Destructor Documentation

##### `__replace_selector()`

```
template<typename _It, typename _Tp>
__gnu_parallel::__replace_selector<_It, _Tp>::__replace_selector (
 const _Tp & __new_val) [inline], [explicit]
```

Constructor.

### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

References `__new_val`.

### 6.53.3 Member Function Documentation

#### `operator>()`

```
template<typename _It, typename _Tp>
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
 _Tp & __v,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Current value.               |
| <code>__i</code> | iterator referencing object. |

References [\\_\\_new\\_val](#).

### 6.53.4 Member Data Documentation

#### `__new_val`

```
template<typename _It, typename _Tp>
const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val
```

Value to replace with.  
Referenced by [\\_\\_replace\\_selector\(\)](#), and [operator>\(\)](#).

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).  
The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.54 `__gnu_cxx::__scoped_lock` Class Reference

```
#include <concurrency.h>
```

#### Public Types

- typedef `__mutex` `__mutex_type`

#### Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

### 6.54.1 Detailed Description

Scoped lock idiom.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 6.55 \_\_gnu\_parallel::\_\_transform1\_selector<\_It> Struct Template Reference

#include <for\_each\_selectors.h>

Inheritance diagram for \_\_gnu\_parallel::\_\_transform1\_selector<\_It>:



### Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 6.55.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform1_selector<_It>
```

`std::transform()` \_\_selector, one input sequence variant.

#### 6.55.2 Member Function Documentation

##### `operator()()`

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform1_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

**Parameters**

|                                        |                              |
|----------------------------------------|------------------------------|
| <a href="#"><code>_↔<br/>_o</code></a> | Operator.                    |
| <a href="#"><code>_↔<br/>_i</code></a> | iterator referencing object. |

**6.55.3 Member Data Documentation****`_M_finish_iterator`**

```
template<typename _It>
```

```
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**6.56 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference**

```
#include <for_each_selectors.h>
```

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:

**Public Member Functions**

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

**Public Attributes**

- `_It` [`\_M\_finish\_iterator`](#)



### 6.56.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform2_selector<_It>
```

`std::transform()` `__selector`, two input sequences variant.

### 6.56.2 Member Function Documentation

#### `operator()`

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform2_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

### 6.56.3 Member Data Documentation

#### `_M_finish_iterator`

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
_iterator on last element processed; needed for some algorithms (e. g. std::transform()).
```

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.57 `__gnu_parallel::__unary_negate<_Predicate, argument_type>` Class Template Reference

```
#include <base.h>
```

Inheritance diagram for `__gnu_parallel::__unary_negate<_Predicate, argument_type >`:



### Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

### Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const argument_type &__x`)

### Protected Attributes

- `_Predicate _M_pred`

#### 6.57.1 Detailed Description

```
template<typename _Predicate, typename argument_type>
class __gnu_parallel::__unary_negate<_Predicate, argument_type >
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

#### 6.57.2 Member Typedef Documentation

##### `argument_type`

```
typedef argument_type std::unary_function< argument_type, bool >::argument_type [inherited]
argument_type is the type of the argument
```

##### `result_type`

```
typedef bool std::unary_function< argument_type, bool >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [base.h](#)

## 6.58 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

```
#include <vstring.h>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **const\_iterator**
- typedef `_CharT_alloc_traits::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **iterator**
- typedef `_CharT_alloc_traits::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_type::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

### Public Member Functions

- `__versa_string` (`__versa_string` &&\_\_str) noexcept
- template<class `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>>`  
`__versa_string` (`_InputIterator` \_\_beg, `_InputIterator` \_\_end, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`const __versa_string &` \_\_str)
- `__versa_string` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n, `const _Alloc &` \_\_a)
- `__versa_string` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n=`npos`)
- `__versa_string` (`const _Alloc &` \_\_a=`_Alloc()`) noexcept
- `__versa_string` (`const _CharT * __s`, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`const _CharT * __s`, `size_type` \_\_n, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`size_type` \_\_n, `_CharT` \_\_c, `const _Alloc &` \_\_a=`_Alloc()`)
- `__versa_string` (`std::initializer_list< _CharT >` \_\_l, `const _Alloc &` \_\_a=`_Alloc()`)
- `~__versa_string` () noexcept
- template<typename `_InputIterator`>  
`__versa_string<_CharT, _Traits, _Alloc, _Base> & _M_replace_dispatch` (`const_iterator` \_\_i1, `const_iterator` \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2, `std::false_type`)
- template<class `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>>`  
`__versa_string & append` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `__versa_string & append` (`const __versa_string &` \_\_str)
- `__versa_string & append` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n)
- `__versa_string & append` (`const _CharT * __s`)
- `__versa_string & append` (`const _CharT * __s`, `size_type` \_\_n)
- `__versa_string & append` (`size_type` \_\_n, `_CharT` \_\_c)
- `__versa_string & append` (`std::initializer_list< _CharT >` \_\_l)
- `__versa_string & assign` (`__versa_string` &&\_\_str) noexcept
- template<class `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>>`  
`__versa_string & assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `__versa_string & assign` (`const __versa_string &` \_\_str)
- `__versa_string & assign` (`const __versa_string &` \_\_str, `size_type` \_\_pos, `size_type` \_\_n)
- `__versa_string & assign` (`const _CharT * __s`)

- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_n)
- [\\_\\_versa\\_string](#) & [assign](#) (size\_type \_\_n, [\\_CharT](#) \_\_c)
- [\\_\\_versa\\_string](#) & [assign](#) (std::initializer\_list< [\\_CharT](#) > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- const\_reference [back](#) () const noexcept
- reference [back](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- iterator [begin](#) () noexcept
- const [\\_CharT](#) \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [\\_\\_versa\\_string](#) & \_\_str) const
- int [compare](#) (const [\\_CharT](#) \* \_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const [\\_\\_versa\\_string](#) & \_\_str) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const [\\_CharT](#) \* \_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const [\\_CharT](#) \* \_\_s, size\_type \_\_n2) const
- int [compare](#) (size\_type \_\_pos1, size\_type \_\_n1, const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- size\_type [copy](#) ([\\_CharT](#) \* \_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- const [\\_CharT](#) \* [data](#) () const noexcept
- bool [empty](#) () const noexcept
- const\_iterator [end](#) () const noexcept
- iterator [end](#) () noexcept
- iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator [erase](#) (const\_iterator \_\_position)
- [\\_\\_versa\\_string](#) & [erase](#) (size\_type \_\_pos=0, size\_type \_\_n=npos)
- size\_type [find](#) ([\\_CharT](#) \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_first\\_not\\_of](#) ([\\_CharT](#) \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_not\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_first\\_not\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_first\\_of](#) ([\\_CharT](#) \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_of](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type [find\\_first\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_first\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos=0) const
- size\_type [find\\_last\\_not\\_of](#) ([\\_CharT](#) \_\_c, size\_type \_\_pos=npo) const noexcept
- size\_type [find\\_last\\_not\\_of](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=npo) const noexcept
- size\_type [find\\_last\\_not\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_not\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos=npo) const
- size\_type [find\\_last\\_of](#) ([\\_CharT](#) \_\_c, size\_type \_\_pos=npo) const noexcept
- size\_type [find\\_last\\_of](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=npo) const noexcept
- size\_type [find\\_last\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [find\\_last\\_of](#) (const [\\_CharT](#) \* \_\_s, size\_type \_\_pos=npo) const

- `const_reference front ()` `const noexcept`
- `reference front ()` `noexcept`
- `allocator_type get_allocator ()` `const noexcept`
- `iterator insert (const_iterator __p, _CharT __c)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)`
- `iterator insert (const_iterator __p, size_type __n, _CharT __c)`
- `iterator insert (const_iterator __p, std::initializer_list<_CharT> __l)`
- `__versa_string & insert (size_type __pos, const _CharT *__s)`
- `__versa_string & insert (size_type __pos, const _CharT *__s, size_type __n)`
- `__versa_string & insert (size_type __pos, size_type __n, _CharT __c)`
- `__versa_string & insert (size_type __pos1, const __versa_string & __str)`
- `__versa_string & insert (size_type __pos1, const __versa_string & __str, size_type __pos2, size_type __n)`
- `size_type length ()` `const noexcept`
- `size_type max_size ()` `const noexcept`
- `__versa_string & operator+= (_CharT __c)`
- `__versa_string & operator+= (const __versa_string & __str)`
- `__versa_string & operator+= (const _CharT *__s)`
- `__versa_string & operator+= (std::initializer_list<_CharT> __l)`
- `__versa_string & operator= (_CharT __c)` `noexcept`
- `__versa_string & operator= (const __versa_string & __str)`
- `__versa_string & operator= (const _CharT *__s)`
- `__versa_string & operator= (std::initializer_list<_CharT> __l)`
- `const_reference operator[] (size_type __pos)` `const noexcept`
- `reference operator[] (size_type __pos)` `noexcept`
- `void pop_back ()`
- `void push_back (_CharT __c)`
- `const_reverse_iterator rbegin ()` `const noexcept`
- `reverse_iterator rbegin ()` `noexcept`
- `const_reverse_iterator rend ()` `const noexcept`
- `reverse_iterator rend ()` `noexcept`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, _CharT *__k1, _CharT *__k2)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`__versa_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const __versa_string & __str)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__k1, const _CharT *__k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s, size_type __n)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, std::initializer_list<_CharT> __l)`
- `__versa_string & replace (size_type __pos, size_type __n, const __versa_string & __str)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `__versa_string & replace (size_type __pos1, size_type __n1, const __versa_string & __str, size_type __pos2, size_type __n2)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n)`

- void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const [\\_\\_versa\\_string](#) & \_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type [rfind](#) (const \_CharT \* \_\_s, size\_type \_\_pos=[npos](#)) const
- void [shrink\\_to\\_fit](#) () noexcept
- size\_type [size](#) () const noexcept
- [\\_\\_versa\\_string](#) substr (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- void [swap](#) ([\\_\\_versa\\_string](#) & \_\_s) noexcept

### Static Public Attributes

- static const size\_type [npos](#)

### 6.58.1 Detailed Description

template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>

class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >

Template class [\\_\\_versa\\_string](#).

Data structure managing sequences of characters and character-like objects.

### 6.58.2 Constructor & Destructor Documentation

#### [\\_\\_versa\\_string\(\)](#) [1/10]

template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>

[\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >::\_\_versa\_string (const \_Alloc & \_\_a = \_Alloc()) [inline], [explicit], [noexcept]

Construct an empty string using allocator *a*.

Referenced by [compare\(\)](#), [compare\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::substr\(\)](#).

#### [\\_\\_versa\\_string\(\)](#) [2/10]

template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>

[\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >::\_\_versa\_string (const [\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base > & \_\_str) [inline]

Construct string with copy of value of *\_\_str*.

### Parameters

|                       |                |
|-----------------------|----------------|
| <a href="#">__str</a> | Source string. |
|-----------------------|----------------|

#### [\\_\\_versa\\_string\(\)](#) [3/10]

template<typename \_CharT, typename \_Traits, typename \_Alloc, template< typename, typename, typename > class \_Base>

[\\_\\_gnu\\_cxx::\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base >::\_\_versa\_string (const [\\_\\_versa\\_string](#)< \_CharT, \_Traits, \_Alloc, \_Base > && \_\_str) [inline], [noexcept]

String move constructor.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

**`__versa_string()` [4/10]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 std::initializer_list< _CharT > __l,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | std::initializer_list of characters.             |
| <code>__a</code> | Allocator to use (default is default allocator). |

**`__versa_string()` [5/10]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n = npos) [inline]
```

Construct string as copy of a substring.

**Parameters**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__str</code> | Source string.                                    |
| <code>__pos</code> | Index of first character to copy from.            |
| <code>__n</code>   | Number of characters to copy (default remainder). |

**`__versa_string()` [6/10]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
```

```

 size_type __n,
 const _Alloc & __a) [inline]

```

Construct string as copy of a substring.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

#### `__versa_string()` [7/10]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const _CharT * __s,
 size_type __n,
 const _Alloc & __a = _Alloc()) [inline]

```

Construct string initialized by a character array.

#### Parameters

|                        |                                                  |
|------------------------|--------------------------------------------------|
| <code>↔<br/>__s</code> | Source character array.                          |
| <code>↔<br/>__n</code> | Number of characters to copy.                    |
| <code>↔<br/>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

#### `__versa_string()` [8/10]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const _CharT * __s,
 const _Alloc & __a = _Alloc()) [inline]

```

Construct string as copy of a C string.

#### Parameters

|                        |                                                  |
|------------------------|--------------------------------------------------|
| <code>↔<br/>__s</code> | Source C string.                                 |
| <code>↔<br/>__a</code> | Allocator to use (default is default allocator). |



**\_\_versa\_string()** [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 size_type __n,
 _CharT __c,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as multiple characters.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__n</code> | Number of characters.                            |
| <code>__c</code> | Character to use.                                |
| <code>__a</code> | Allocator to use (default is default allocator). |

**\_\_versa\_string()** [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 _InputIterator __beg,
 _InputIterator __end,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a range.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

**~\_\_versa\_string()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string () [inline], [noexcept]
```

Destroy the string instance.

**6.58.3 Member Function Documentation****append()** [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Append a range of characters.

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

#### Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

#### append() [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Append a string to this string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

#### Returns

Reference to this string.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::append\(\)](#), [std::getline\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator+=\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator+=\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator+=\(\)](#), and [resize\(\)](#).

#### append() [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n) [inline]
```

Append a substring.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | The string to append.                          |
| <code>__pos</code> | Index of the first character of str to append. |
| <code>__n</code>   | The number of characters to append.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                     |
|------------------------------------------------|-------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | if <i>pos</i> is not a valid index. |
|------------------------------------------------|-------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
 const _CharT * __s,
 size_type __n) [inline]
```

Append a C substring.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

**Returns**

Reference to this string.

**append()** [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 size_type __n,
 _CharT __c) [inline]
```

Append multiple characters.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__n</code> | The number of characters to append. |
| <code>__c</code> | The character to use.               |

#### Returns

Reference to this string.

Appends n copies of c to this string.

#### append() [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 std::initializer_list< _CharT > __l) [inline]
```

Append an initializer\_list of characters.

#### Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__l</code> | The initializer_list of characters to append. |
|------------------|-----------------------------------------------|

#### Returns

Reference to this string.

#### assign() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 __versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

Set value to contents of another string.

#### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

**assign()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::operator=()`, `__gnu_cxx::__versa_string<char>::operator=()`, `__gnu_cxx::__versa_string<char>::operator=()`, and `__gnu_cxx::__versa_string<char>::operator=()`.

**assign()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n) [inline]
```

Set value to a substring of a string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                             |
|------------------------------------------------|---------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | if <code>__pos</code> is not a valid index. |
|------------------------------------------------|---------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

#### **assign()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const _CharT * __s) [inline]
```

Set value to contents of a C string.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

#### Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

#### **assign()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const _CharT * __s,
 size_type __n) [inline]
```

Set value to a C substring.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
 size_type __n,
 _CharT __c) [inline]
```

Set value to multiple characters.

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**assign()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
 std::initializer_list<_CharT > __l) [inline]
```

Set value to an `initializer_list` of characters.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__l</code> | The <code>initializer_list</code> of characters to assign. |
|------------------|------------------------------------------------------------|

**Returns**

Reference to this string.

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
 size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                                                                                    |                                       |
|----------------------------------------------------------------------------------------------------|---------------------------------------|
|  <code>__n</code> | The index of the character to access. |
|----------------------------------------------------------------------------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                                                                                      |                                       |
|------------------------------------------------------------------------------------------------------|---------------------------------------|
|  <code>__n</code> | The index of the character to access. |
|------------------------------------------------------------------------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.



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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

**back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**begin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Referenced by `__gnu_cxx::__versa_string< char >::crend()`, `__gnu_cxx::__versa_string< char >::rend()`, and `__gnu_cxx::__versa_string< char >::rend()`.

**c\_str()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

**capacity()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by `__gnu_cxx::__versa_string< char >::push_back()`, and `__gnu_cxx::__versa_string< char >::shrink_to_fit()`.

**cbegin()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cend()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const [inline],
[noexcept]
```

## clear()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear () [inline], [noexcept]
Erases the string, making it empty.
```

## compare() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const [inline]
```

## Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>()`, `__gnu_cxx::operator>=()`, `__gnu_cxx::operator>=()`, and `__gnu_cxx::operator>=()`.

## compare() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 const _CharT * __s) const
```

## Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_↵</code> | C string to compare against. |
| <code>_s</code> |                              |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References [std::min\(\)](#), and [size\(\)](#).

**compare()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References [\\_\\_versa\\_string\(\)](#), [data\(\)](#), [std::min\(\)](#), and [size\(\)](#).

**compare()** [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References [std::min\(\)](#).

**compare()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const
```

Compare substring against a character array.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `\0` has no special meaning.

References [std::min\(\)](#).

**compare()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos1,
```

```

size_type __n1,
const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,
size_type __pos2,
size_type __n2) const

```

Compare substring to a substring.

#### Parameters

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(), str.substr(pos2, n2).data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

References `__versa_string()`, `data()`, and `std::min()`.

#### copy()

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, ↵
_Traits, _Alloc, _Base>::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const

```

Copy substring into C string.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

**Returns**

Number of characters actually copied

**Exceptions**

|                                                |                                   |
|------------------------------------------------|-----------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos &gt; size()</code> . |
|------------------------------------------------|-----------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**crbegin()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crend()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**data()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data () const [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by [compare\(\)](#), [compare\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::compare\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_not\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::operator==\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_operator==\(\)](#).

**empty()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**end()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end\(\) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::crbegin\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rbegin\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::rbegin\(\)](#).

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase (
 const_iterator __position) [inline]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase (
```

```
size_type __pos = 0,
size_type __n = npos) [inline]
```

Remove characters.

### Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

### Returns

Reference to this string.

### Exceptions

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |
|------------------------------------------------|---------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are  $< __n$  characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown. Referenced by [`std::getline\(\)`](#).

### `find()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a character.

### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`. References [`npos`](#), and [`size\(\)`](#).

### `find()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
```



```
const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,
size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `find()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, ↵
_Traits, _Alloc, _Base>::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a C substring.

#### Parameters

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                       |
| <code>__pos</code> | Index of character to search from.                        |
| <code>__n</code>   | Number of characters from <code>__s</code> to search for. |

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<](#)

#### `find()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find\_first\_not\_of() [1/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_first\_not\_of() [2/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_first\\_not\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::find\\_first\\_not\\_of\(\)](#).

**find\_first\_not\_of() [3/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search from.                    |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_first\_not\_of() [4/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of() [1/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

**find\_first\_of() [2/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_first\\_of\(\)](#).

**find\_first\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_first\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_traits, _Alloc, _Base >::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

#### `find_last_not_of()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_last\\_not\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_last\\_not\\_of\(\)](#).

#### `find_last_not_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_traits, _Alloc, _Base >::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_last\_not\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

**find\_last\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_last\\_of\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_last\\_of\(\)](#).

**find\_last\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |



**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

**find\_last\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

**get\_allocator()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ()
const [inline], [noexcept]
```

Return copy of allocator used to construct this string.

**insert()** [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 _CharT __c) [inline]
```

Insert one character.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <a href="#">__p</a> | Iterator referencing position in string to insert at. |
| <a href="#">__c</a> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                   |                                                 |
|-----------------------------------|-------------------------------------------------|
| <a href="#">std::length_error</a> | If new length exceeds <code>max_size()</code> . |
|-----------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline]
```

Insert a range of characters.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                              |
| <code>__c</code> | The character to insert.                                    |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown. Referenced by `__gnu_cxx::__versa_string< char >::insert()`.

**insert()** [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 std::initializer_list< _CharT > __l) [inline]
```

Insert an `initializer_list` of characters.

**Parameters**

|                                        |                                                             |
|----------------------------------------|-------------------------------------------------------------|
| <a href="#"><code>_↵<br/>_p</code></a> | Const_iterator referencing location in string to insert at. |
| <a href="#"><code>_↵<br/>_l</code></a> | The initializer_list of characters to insert.               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

**insert()** [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 const _CharT * __s) [inline]
```

Insert a C string.

**Parameters**

|                                    |                                                       |
|------------------------------------|-------------------------------------------------------|
| <a href="#"><code>__pos</code></a> | Iterator referencing location in string to insert at. |
| <a href="#"><code>__s</code></a>   | The C string to insert.                               |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n) [inline]
```

Insert a C substring.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |
| <code>__n</code>   | The number of characters to insert.                   |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

#### `insert()` [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Insert value of a string.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos2,
 size_type __n) [inline]
```

Insert a substring.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert.    |
| <code>__n</code>    | Number of characters to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                                               |
|------------------------------------------------|-------------------------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                               |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const [inline],
[noexcept]
```

Returns the `size()` of the largest possible string.

Referenced by [`std::getline\(\)`](#).

**operator+=() [1/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 _CharT __c) [inline]
```

Append a character.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

**Returns**

Reference to this string.

**operator+=()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator+=()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**operator+=()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 std::initializer_list< _CharT > __l) [inline]
```

Append an `initializer_list` of characters.



**Parameters**

|                |                                                    |
|----------------|----------------------------------------------------|
| <code>↩</code> | The initializer_list of characters to be appended. |
| <code>↩</code> |                                                    |
| <code>↩</code> |                                                    |
| <code>↩</code> |                                                    |
| <code>/</code> |                                                    |

**Returns**

Reference to this string.

**operator=()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
 __versa_string<_CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

String move assignment operator.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

**operator=()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
 _CharT __c) [inline]
```

Set value to string of length 1.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>↩</code>  | Source character. |
| <code>_c</code> |                   |

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

**operator=()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (
 const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Assign the value of `str` to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**operator=()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 const _CharT * __s) [inline]
```

Copy contents of `__s` into this string.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

**operator=()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 std::initializer_list< _CharT > __l) [inline]
```

Set value to string constructed from initializer list.

**Parameters**

|                                    |  |
|------------------------------------|--|
| <code>std::initializer_list</code> |  |
|------------------------------------|--|

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
 size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by `__gnu_cxx::__versa_string< char >::back()`, `__gnu_cxx::__versa_string< char >::back()`, `__gnu_cxx::__versa_string< char >::front()` and `__gnu_cxx::__versa_string< char >::front()`.

**operator[]()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
 size_type __pos) [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

**pop\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

**push\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (
 _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to append. |
|------------------|----------------------|

Referenced by [\\_\\_gnu\\_cxx::operator+\(\)](#), [\\_\\_gnu\\_cxx::operator+\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<char>::operator+=\(\(\)\)](#).

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

#### rend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () const
[inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

#### rend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

#### replace() [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline]
```

Replace range of characters with range.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### `replace()` [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Replace range of characters with string.

#### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### `replace()` [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const _CharT * __s) [inline]
```

Replace range of characters with C string.

#### Parameters

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↵<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↵<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↵<br/>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const _CharT * __s,
 size_type __n) [inline]
```

Replace range of characters with C substring.

**Parameters**

|                                   |                                                     |
|-----------------------------------|-----------------------------------------------------|
| <a href="#"><code>__i1</code></a> | Iterator referencing start of range to replace.     |
| <a href="#"><code>__i2</code></a> | Iterator referencing end of range to replace.       |
| <a href="#"><code>__s</code></a>  | C string value to insert.                           |
| <a href="#"><code>__n</code></a>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
```

```
size_type __n,
_CharT __c) [inline]
```

Replace range of characters with multiple characters.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### `replace()` [6/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 std::initializer_list<_CharT > __l) [inline]
```

Replace range of characters with `initializer_list`.

#### Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Replace characters with value from another string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `__gnu_cxx::__versa_string< char >::append()`, `__gnu_cxx::__versa_string< char >::assign()`, `__gnu_cxx::__versa_string< char >::insert()`, `__gnu_cxx::__versa_string< char >::insert()`, `__gnu_cxx::__versa_string< char >::insert()`, `__gnu_cxx::__versa_string< char >::insert()`, `__gnu_cxx::__versa_string< char >::insert()`, `__gnu_cxx::__versa_string< char >::replace()`, `__gnu_cxx::__versa_string< char >::replace()`, `__gnu_cxx::__versa_string< char >::replace()`, `__gnu_cxx::__versa_string< char >::replace()`, `__gnu_cxx::__versa_string< char >::replace()`, `__gnu_cxx::__versa_string< char >::replace()`, and `__gnu_cxx::__versa_string< char >::replace()`.

**replace()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```



```

__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline]

```

Replace characters with value of a C string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> .             |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### `replace()` [9/11]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) [inline]

```

Replace characters with value of a C substring.

#### Parameters

|                    |                                                    |
|--------------------|----------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.               |
| <code>__n1</code>  | Number of characters to be replaced.               |
| <code>__s</code>   | C string to insert.                                |
| <code>__n2</code>  | Number of characters from <code>__s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos1 &gt; size()</code> .            |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n1,
 size_type __n2,
 _CharT __c) [inline]
```

Replace characters with multiple characters.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> .             |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos1,
```

```

size_type __n1,
const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str,
size_type __pos2,
size_type __n2) [inline]

```

Replace characters with value from another string.

### Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

### Returns

Reference to this string.

### Exceptions

|                                                |                                                                             |
|------------------------------------------------|-----------------------------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                             |

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### reserve()

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve (
 size_type __res_arg = 0) [inline]

```

Attempt to preallocate enough memory for specified number of characters.

### Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

### Exceptions

|                                                |                                                             |
|------------------------------------------------|-------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Referenced by [`\_\_gnu\_cxx::operator+\(\)`](#), [`\_\_gnu\_cxx::operator+\(\)`](#), [`\_\_gnu\_cxx::operator+\(\)`](#), and [`\_\_gnu\_cxx::\_\_versa\_string<char>::shrink`](#)

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
 size_type __n) [inline]
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

**resize()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
 size_type __n,
 _CharT __c)
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

References [append\(\)](#), and [size\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::resize\(\)](#).

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [size\(\)](#).

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::find\\_last\\_of\(\)](#), [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::rfind\(\)](#), and [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< char >::rfind\(\)](#).

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

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |



`__gnu_cxx::__versa_string<char>::pop_back()`, `__gnu_cxx::__versa_string<char>::push_back()`, `__gnu_cxx::__versa_string<char>::replace()`, `resize()`, `rfind()`, `rfind()`, and `__gnu_cxx::__versa_string<char>::shrink_to_fit()`.

### substr()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline]
```

Get a substring.

#### Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

#### Returns

The new string.

#### Exceptions

|                                |                                   |
|--------------------------------|-----------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> . |
|--------------------------------|-----------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

### swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::swap (
 __versa_string<_CharT, _Traits, _Alloc, _Base> & __s) [inline], [noexcept]
```

Swap contents with another string.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

Referenced by `__gnu_cxx::__versa_string<char>::assign()`, `__gnu_cxx::__versa_string<char>::operator=()`, and `__gnu_cxx::swap()`.

#### 6.58.4 Member Data Documentation

##### npos

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos [static]
```

Value returned by various member functions when they fail.

Referenced by [find\(\)](#), [find\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_of\(\)](#), [rfind\(\)](#), and [rfind\(\)](#).

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

## 6.59 `__gnu_debug::After_nth_from<_Iterator>` Class Template Reference

```
#include <safe_sequence.h>
```

### Public Member Functions

- constexpr `After_nth_from` (const difference\_type &\_\_n, const \_Iterator &\_\_base)
- bool `operator()` (const \_Iterator &\_\_x) const

#### 6.59.1 Detailed Description

```
template<typename _Iterator>
class __gnu_debug::After_nth_from<_Iterator>
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 6.60 `std::Base_bitset<_Nw>` Struct Template Reference

```
#include <bitset>
```

### Public Types

- typedef unsigned long `_WordT`

### Public Member Functions

- constexpr `Base_bitset` (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
constexpr bool `_M_are_all` () const noexcept
- constexpr void `_M_do_and` (const Base\_bitset<\_Nw> &\_\_x) noexcept
- constexpr size\_t `_M_do_count` () const noexcept
- constexpr size\_t `_M_do_find_first` (size\_t) const noexcept
- constexpr size\_t `_M_do_find_next` (size\_t, size\_t) const noexcept
- constexpr void `_M_do_flip` () noexcept
- constexpr void `_M_do_left_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_or` (const Base\_bitset<\_Nw> &\_\_x) noexcept
- constexpr void `_M_do_reset` () noexcept
- constexpr void `_M_do_right_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_set` () noexcept
- constexpr unsigned long long `_M_do_to_ullong` () const
- constexpr unsigned long `_M_do_to_ulong` () const
- constexpr void `_M_do_xor` (const Base\_bitset<\_Nw> &\_\_x) noexcept
- constexpr const \_WordT \* `_M_getdata` () const noexcept
- constexpr \_WordT `_M_getword` (size\_t \_\_pos) const noexcept



- constexpr `_WordT & _M_getword (size_t __pos)` noexcept
- constexpr `_WordT _M_hiword ()` const noexcept
- constexpr `_WordT & _M_hiword ()` noexcept
- constexpr `bool _M_is_any ()` const noexcept
- constexpr `bool _M_is_equal (const _Base_bitset< _Nw > &__x)` const noexcept

### Static Public Member Functions

- static constexpr `_WordT _S_maskbit (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbit (size_t __pos)` noexcept
- static constexpr `size_t _S_whichbyte (size_t __pos)` noexcept
- static constexpr `size_t _S_whichword (size_t __pos)` noexcept

### Public Attributes

- `_WordT _M_w [_Nw]`

#### 6.60.1 Detailed Description

`template<size_t _Nw>`

`struct std::_Base_bitset< _Nw >`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative. See documentation for `bitset`.

#### 6.60.2 Member Data Documentation

##### `_M_w`

`template<size_t _Nw>`

`_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 6.61 `std::_Base_bitset< 0 >` Struct Reference

```
#include <bitset>
```

### Public Types

- typedef unsigned long `_WordT`

### Public Member Functions

- constexpr `_Base_bitset (unsigned long long)` noexcept
- `template<size_t _Nb>`  
constexpr `bool _M_are_all ()` const noexcept
- constexpr `void _M_do_and (const _Base_bitset< 0 > &)` noexcept
- constexpr `size_t _M_do_count ()` const noexcept
- constexpr `size_t _M_do_find_first (size_t)` const noexcept
- constexpr `size_t _M_do_find_next (size_t, size_t)` const noexcept
- constexpr `void _M_do_flip ()` noexcept
- constexpr `void _M_do_left_shift (size_t)` noexcept

- constexpr void **\_M\_do\_or** (const \_Base\_bitset< 0 > &) noexcept
- constexpr void **\_M\_do\_reset** () noexcept
- constexpr void **\_M\_do\_right\_shift** (size\_t) noexcept
- constexpr void **\_M\_do\_set** () noexcept
- constexpr unsigned long long **\_M\_do\_to\_ullong** () const noexcept
- constexpr unsigned long **\_M\_do\_to\_ulong** () const noexcept
- constexpr void **\_M\_do\_xor** (const \_Base\_bitset< 0 > &) noexcept
- constexpr const \_WordT \* **\_M\_getdata** () const noexcept
- constexpr \_WordT **\_M\_getword** (size\_t) const noexcept
- \_WordT & **\_M\_getword** (size\_t) noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- constexpr bool **\_M\_is\_any** () const noexcept
- constexpr bool **\_M\_is\_equal** (const \_Base\_bitset< 0 > &) const noexcept

### Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

### Public Attributes

- \_WordT **\_M\_w** [*Nw*]

#### 6.61.1 Detailed Description

Base class, specialization for no storage (zero-length bitset).  
See documentation for bitset.

#### 6.61.2 Member Data Documentation

##### **\_M\_w**

\_WordT `std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

The documentation for this struct was generated from the following file:

- [bitset](#)

## 6.62 std::\_Base\_bitset< 1 > Struct Reference

```
#include <bitset>
```

### Public Types

- typedef unsigned long **\_WordT**

### Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
constexpr bool **\_M\_are\_all** () const noexcept
- constexpr void **\_M\_do\_and** (const \_Base\_bitset< 1 > &\_\_x) noexcept
- constexpr size\_t **\_M\_do\_count** () const noexcept

- constexpr size\_t `_M_do_find_first` (size\_t \_\_not\_found) const noexcept
- constexpr size\_t `_M_do_find_next` (size\_t \_\_prev, size\_t \_\_not\_found) const noexcept
- constexpr void `_M_do_flip` () noexcept
- constexpr void `_M_do_left_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_or` (const \_Base\_bitset< 1 > &\_\_x) noexcept
- constexpr void `_M_do_reset` () noexcept
- constexpr void `_M_do_right_shift` (size\_t \_\_shift) noexcept
- constexpr void `_M_do_set` () noexcept
- constexpr unsigned long long `_M_do_to_ullong` () const noexcept
- constexpr unsigned long `_M_do_to_ulong` () const noexcept
- constexpr void `_M_do_xor` (const \_Base\_bitset< 1 > &\_\_x) noexcept
- constexpr const \_WordT \* `_M_getdata` () const noexcept
- constexpr \_WordT `_M_getword` (size\_t) const noexcept
- constexpr \_WordT & `_M_getword` (size\_t) noexcept
- constexpr \_WordT `_M_hiword` () const noexcept
- constexpr \_WordT & `_M_hiword` () noexcept
- constexpr bool `_M_is_any` () const noexcept
- constexpr bool `_M_is_equal` (const \_Base\_bitset< 1 > &\_\_x) const noexcept

#### Static Public Member Functions

- static constexpr \_WordT `_S_maskbit` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichbit` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichbyte` (size\_t \_\_pos) noexcept
- static constexpr size\_t `_S_whichword` (size\_t \_\_pos) noexcept

#### Public Attributes

- \_WordT `_M_w`

#### 6.62.1 Detailed Description

Base class, specialization for a single word.

See documentation for bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

### 6.63 `__gnu_debug::_BeforeBeginHelper<_Sequence>` Struct Template Reference

```
#include <safe_iterator.h>
```

#### Static Public Member Functions

- template<typename \_Iterator, typename \_Category>  
static bool `_S_Is` (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &)
- template<typename \_Iterator, typename \_Category>  
static bool `_S_Is_Beginnest` (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &\_\_it)

### 6.63.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper< _Sequence >
```

Helper struct to deal with sequence offering a `before_begin` iterator.  
The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 6.64 `std::_Bind< _Signature >` Class Template Reference

### 6.64.1 Detailed Description

```
template<typename _Signature>
class std::_Bind< _Signature >
```

Type of the function object returned from `bind()`.  
The documentation for this class was generated from the following file:

- [functional](#)

## 6.65 `std::_Bind_result< _Result, _Signature >` Class Template Reference

### 6.65.1 Detailed Description

```
template<typename _Result, typename _Signature>
class std::_Bind_result< _Result, _Signature >
```

Type of the function object returned from `bind<R>()`.  
The documentation for this class was generated from the following file:

- [functional](#)

## 6.66 `__gnu_cxx::__detail::_Bitmap_counter< _Tp >` Class Template Reference

```
#include <bitmap_allocator.h>
```

### Public Member Functions

- `_Bitmap_counter` (`_BPVector` &`Rvbp`, `long` `__index=-1`)
- `pointer` `_M_base` () const throw ()
- `bool` `_M_finished` () const throw ()
- `std::size_t` \* `_M_get` () const throw ()
- `_Index_type` `_M_offset` () const throw ()
- `void` `_M_reset` (`long` `__index=-1`) throw ()
- `void` `_M_set_internal_bitmap` (`std::size_t` \* `__new_internal_marker`) throw ()
- `_Index_type` `_M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

### 6.66.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter< _Tp >
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.  
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 6.67 `std::__detail::_BracketMatcher<_TraitsT, __icase, __collate >` Struct Template Reference

```
#include <regex_compiler.h>
```

### Public Types

- `typedef _TraitsT::char_class_type _CharClassT`
- `typedef _TransT::_CharT _CharT`
- `typedef _TraitsT::string_type _StringT`
- `typedef _TransT::_StrTransT _StrTransT`
- `typedef _RegexTranslator<_TraitsT, __icase, __collate > _TransT`

### Public Member Functions

- `_BracketMatcher` (bool \_\_is\_non\_matching, const \_TraitsT &\_\_traits)
- `void _M_add_char` (\_CharT \_\_c)
- `void _M_add_character_class` (const \_StringT &\_\_s, bool \_\_neg)
- `_StringT _M_add_collate_element` (const \_StringT &\_\_s)
- `void _M_add_equivalence_class` (const \_StringT &\_\_s)
- `void _M_make_range` (\_CharT \_\_l, \_CharT \_\_r)
- `void _M_ready` ()
- `bool operator()` (\_CharT \_\_ch) const

#### 6.67.1 Detailed Description

```
template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher<_TraitsT, __icase, __collate >
```

Matches a character range (bracket expression).

The documentation for this struct was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

## 6.68 `__gnu_cxx::_Caster<_ToType >` Struct Template Reference

```
#include <cast.h>
```

### Public Types

- `typedef _ToType::element_type * type`

#### 6.68.1 Detailed Description

```
template<typename _ToType>
struct __gnu_cxx::_Caster<_ToType >
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

The documentation for this struct was generated from the following file:

- [cast.h](#)

## 6.69 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

```
#include <char_traits.h>
```

### Public Types

- typedef unsigned long `int_type`
- typedef `std::streamoff` `off_type`
- typedef `std::streampos` `pos_type`
- typedef `std::mbstate_t` `state_type`

### 6.69.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::_Char_types<_CharT>
```

Mapping from character type to associated types.

#### Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 6.70 `__gnu_pbds::detail::pat_trie_base::_Clter<Node, Leaf, Head, Inode, Is_Forward_Iterator>` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



## Public Types

- typedef allocator\_type **\_Alloc**
- typedef Node::allocator\_type **allocator\_type**
- typedef type\_traits::const\_pointer **const\_pointer**
- typedef type\_traits::const\_reference **const\_reference**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef [rebind\\_traits](#)< \_Alloc, Head >::pointer **head\_pointer**
- typedef Inode::iterator **inode\_iterator**
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::pointer **inode\_pointer**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::pointer **leaf\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Node >::pointer **node\_pointer**
- typedef type\_traits::pointer **pointer**
- typedef type\_traits::reference **reference**
- typedef Node::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

## Public Member Functions

- **\_CIter** (const \_CIter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- **\_CIter** (node\_pointer p\_nd=0)
- bool **operator!=** (const \_CIter &other) const
- bool **operator!=** (const \_CIter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- const\_reference **operator\*** () const

- `_Clter & operator++ ()`
- `_Clter operator++ (int)`
- `_Clter & operator-- ()`
- `_Clter operator-- (int)`
- `const_pointer operator-> () const`
- `_Clter & operator= (const _Clter &other)`
- `_Clter & operator= (const _Clter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)`
- `bool operator== (const _Clter &other) const`
- `bool operator== (const _Clter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const`

### Public Attributes

- node\_pointer `m_p_nd`

### Protected Member Functions

- void `dec` (false\_type)
- void `dec` (true\_type)
- void `inc` (false\_type)
- void `inc` (true\_type)

### Static Protected Member Functions

- static node\_pointer `get_larger_sibling` (node\_pointer p\_nd)
- static node\_pointer `get_smaller_sibling` (node\_pointer p\_nd)
- static leaf\_pointer `leftmost_descendant` (node\_pointer p\_nd)
- static leaf\_pointer `rightmost_descendant` (node\_pointer p\_nd)

#### 6.70.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 6.71 `std::__detail::_Compiler< _TraitsT >` Class Template Reference

```
#include <regex_compiler.h>
```

### Public Types

- typedef `_TraitsT::char_type` `_CharT`
- typedef [regex\\_constants::syntax\\_option\\_type](#) `_FlagT`
- typedef `_NFA< _TraitsT >` `_RegexT`

### Public Member Functions

- `_Compiler` (const `_CharT` \*\_\_b, const `_CharT` \*\_\_e, const typename `_TraitsT::locale_type` &\_\_traits, `_FlagT` \_\_flags)
- [shared\\_ptr](#)< const `_RegexT` > `_M_get_nfa` () noexcept



### 6.71.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_Compiler<_TraitsT >
```

Builds an NFA from an input iterator range.

The `_TraitsT` type should fulfill the requirements in C++11 28.3 [re.req].

The documentation for this class was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

## 6.72 std::\_\_parallel::\_CRandNumber<\_MustBeInt > Struct Template Reference

```
#include <algo.h>
```

### Public Member Functions

- `int operator() (int __limit)`

### 6.72.1 Detailed Description

```
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber<_MustBeInt >
```

Functor wrapper for `std::rand()`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

## 6.73 std::\_Deque\_base<\_Tp, \_Alloc > Class Template Reference

```
#include <stl_deque.h>
```

Inheritance diagram for `std::_Deque_base<_Tp, _Alloc >`:



### Protected Types

- `enum { _S_initial_map_size }`
- `typedef __gnu_cxx::__alloc_traits<_Tp_alloc_type > _Alloc_traits`
- `typedef __gnu_cxx::__alloc_traits<_Map_alloc_type > _Map_alloc_traits`

- typedef `_Alloc_traits::template rebind<_Ptr>::other` **\_Map\_alloc\_type**
- typedef `iterator::_Map_pointer` **\_Map\_pointer**
- typedef `_Alloc_traits::pointer` **\_Ptr**
- typedef `_Alloc_traits::const_pointer` **\_Ptr\_const**
- typedef `__gnu_cxx::__alloc_traits<_Alloc>::template rebind<_Tp>::other` **\_Tp\_alloc\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `_Deque_iterator<_Tp, const _Tp &, _Ptr_const>` **const\_iterator**
- typedef `_Deque_iterator<_Tp, _Tp &, _Ptr>` **iterator**

### Protected Member Functions

- **\_Deque\_base** (`_Deque_base &&__x`)
- **\_Deque\_base** (`_Deque_base &&__x, const allocator_type &__a`)
- **\_Deque\_base** (`_Deque_base &&__x, const allocator_type &__a, size_t __n`)
- **\_Deque\_base** (`const allocator_type &__a`)
- **\_Deque\_base** (`const allocator_type &__a, size_t __num_elements`)
- **\_Deque\_base** (`size_t __num_elements`)
- `_Map_pointer` **\_M\_allocate\_map** (`size_t __n`)
- `_Ptr` **\_M\_allocate\_node** ()
- void **\_M\_create\_nodes** (`_Map_pointer __nstart, _Map_pointer __nfinish`)
- void **\_M\_deallocate\_map** (`_Map_pointer __p, size_t __n`) noexcept
- void **\_M\_deallocate\_node** (`_Ptr __p`) noexcept
- void **\_M\_destroy\_nodes** (`_Map_pointer __nstart, _Map_pointer __nfinish`) noexcept
- `_Map_alloc_type` **\_M\_get\_map\_allocator** () const noexcept
- `const _Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () const noexcept
- `_Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () noexcept
- void **\_M\_initialize\_map** (`size_t`)
- `allocator_type` **get\_allocator** () const noexcept

### Protected Attributes

- `_Deque_impl` **\_M\_impl**

#### 6.73.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_Deque_base<_Tp, _Alloc>
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

#### 6.73.2 Member Function Documentation

##### **\_M\_initialize\_map()**

```
template<typename _Tp, typename _Alloc>
void std::_Deque_base<_Tp, _Alloc>::_M_initialize_map (
 size_t __num_elements) [protected]
```

Layout storage.

**Parameters**

|                             |                                                        |
|-----------------------------|--------------------------------------------------------|
| <code>__num_elements</code> | The count of T's for which to allocate space at first. |
|-----------------------------|--------------------------------------------------------|

The initial underlying memory layout is a bit complicated...

References [std::max\(\)](#).

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

**6.74 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference**

```
#include <stl_deque.h>
```

**Public Types**

- typedef `__ptr_rebind<_Ptr, _Tp>` **Elt\_pointer**
- typedef `__ptr_rebind<_Ptr, Elt_pointer>` **Map\_pointer**
- typedef `_Deque_iterator` **Self**
- typedef `__iter<const _Tp>` **const\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__iter<_Tp>` **iterator**
- typedef `std::random_access_iterator_tag` **iterator\_category**
- typedef `_Ptr` **pointer**
- typedef `_Ref` **reference**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- `_Deque_iterator` (`_Elt_pointer __x, _Map_pointer __y`) noexcept
- `_Deque_iterator` (`const _Deque_iterator &__x`) noexcept
- `template<typename _Iter, typename = _Require<is_same<_Self, const_iterator>, is_same<_Iter, iterator>>>`  
`_Deque_iterator` (`const _Iter &__x`) noexcept
- `iterator _M_const_cast` () const noexcept
- `void _M_set_node` (`_Map_pointer __new_node`) noexcept
- `reference operator*` () const noexcept
- `_Self & operator++` () noexcept
- `_Self operator++` (`int`) noexcept
- `_Self & operator+=` (`difference_type __n`) noexcept
- `_Self & operator--` () noexcept
- `_Self operator--` (`int`) noexcept
- `_Self & operator-=` (`difference_type __n`) noexcept
- `pointer operator->` () const noexcept
- `_Deque_iterator & operator=` (`const _Deque_iterator &`)=default
- `reference operator[]` (`difference_type __n`) const noexcept

**Static Public Member Functions**

- `static size_t _S_buffer_size` () noexcept

## Public Attributes

- `_Elt_pointer _M_cur`
- `_Elt_pointer _M_first`
- `_Elt_pointer _M_last`
- `_Map_pointer _M_node`

## Friends

- `_Self operator+ (const _Self &__x, difference_type __n) noexcept`
- `_Self operator+ (difference_type __n, const _Self &__x) noexcept`
- `template<typename _RefR, typename _PtrR>  
difference_type operator- (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `difference_type operator- (const _Self &__x, const _Self &__y) noexcept`
- `_Self operator- (const _Self &__x, difference_type __n) noexcept`
- `strong_ordering operator<=> (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR>  
bool operator== (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator== (const _Self &__x, const _Self &__y) noexcept`

### 6.74.1 Detailed Description

`template<typename _Tp, typename _Ref, typename _Ptr>  
struct std::_Deque_iterator< _Tp, _Ref, _Ptr >`

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

### 6.74.2 Member Function Documentation

#### `_M_set_node()`

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
 _Map_pointer __new_node) [inline], [noexcept]
```

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

The documentation for this struct was generated from the following files:

- [stl\\_algobase.h](#)
- [stl\\_deque.h](#)

## 6.75 `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >` Struct Template Reference

```
#include <random_shuffle.h>
```

### Public Types

- `typedef _TraitsType::difference_type DifferenceType`
- `typedef std::iterator\_traits< _RAIter > TraitsType`
- `typedef _TraitsType::value_type ValueType`

## Public Member Functions

- [\\_DRandomShufflingGlobalData](#) ([\\_RAIter](#) & [\\_\\_source](#))

## Public Attributes

- [\\_ThreadIndex](#) \* [\\_M\\_bin\\_proc](#)
- [\\_DifferenceType](#) \*\* [\\_M\\_dist](#)
- [int](#) [\\_M\\_num\\_bins](#)
- [int](#) [\\_M\\_num\\_bits](#)
- [\\_RAIter](#) & [\\_M\\_source](#)
- [\\_DifferenceType](#) \* [\\_M\\_starts](#)
- [\\_ValueType](#) \*\* [\\_M\\_temporaries](#)

### 6.75.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>
```

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

### 6.75.2 Constructor & Destructor Documentation

#### `_DRandomShufflingGlobalData()`

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_DRandomShufflingGlobalData (
 _RAIter & __source) [inline]
```

Constructor.

References [\\_M\\_source](#).

### 6.75.3 Member Data Documentation

#### `_M_bin_proc`

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.

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

#### `_M_dist`

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.

Dimensions (`_M_num_threads + 1`) `__x` (`_M_num_bins + 1`).

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

#### `_M_num_bins`

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_num_bins
```

Number of bins to distribute to.

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

### **`_M_num_bits`**

```
template<typename _RAIter>
```

```
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.

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

### **`_M_source`**

```
template<typename _RAIter>
```

```
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the `__source`.

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

### **`_M_starts`**

```
template<typename _RAIter>
```

```
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' `__chunks`.

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

### **`_M_temporaries`**

```
template<typename _RAIter>
```

```
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries
```

Temporary arrays for each thread.

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

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## **6.76 `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >` Struct Template Reference**

```
#include <random_shuffle.h>
```

### **Public Attributes**

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData< \\_RAIter > \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

#### **6.76.1 Detailed Description**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >
```

Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).

#### **6.76.2 Member Data Documentation**

##### **`__bins_end`**

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end
```

End index for bins taken care of by this thread.

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

### `_M_bins_begin`

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
_BinIndex __gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin
```

Begin index for bins taken care of by this thread.

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

### `_M_num_threads`

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
int __gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads
```

Number of threads participating in total.

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

### `_M_sd`

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
_DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd
```

Pointer to global data.

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

### `_M_seed`

```
template<typename _RAIter, typename _RandomNumberGenerator>
```

```
uint32_t __gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed
```

Random `_M_seed` for this thread.

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

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 6.77 `__gnu_parallel::__DummyReduct` Struct Reference

```
#include <for_each_selectors.h>
```

### Public Member Functions

- `bool operator()` (`bool`, `bool`) `const`

#### 6.77.1 Detailed Description

Reduction function doing nothing.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.78 `__gnu_debug::__Equal_to`<`_Type`> Class Template Reference

```
#include <safe_sequence.h>
```

## Public Member Functions

- constexpr **Equal\_to** (const \_Type &\_\_v)
- bool **operator()** (const \_Type &\_\_x) const

### 6.78.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::Equal_to< _Type >
```

A simple function object that returns true if the passed-in value is equal to the stored value.  
The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 6.79 \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare >:



## Public Types

- typedef \_T1 [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_T2 [second\\_argument\\_type](#)

## Public Member Functions

- **EqualFromLess** (\_Compare &\_\_comp)
- bool **operator()** (const \_T1 &\_\_a, const \_T2 &\_\_b)

### 6.79.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.



### 6.79.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

#### result\_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]
```

result\_type is the return type

#### second\_argument\_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [base.h](#)

## 6.80 \_\_gnu\_parallel::\_EqualTo< \_T1, \_T2 > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_EqualTo< \_T1, \_T2 >:



### Public Types

- typedef \_T1 [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_T2 [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_T1 &\_\_t1, const \_T2 &\_\_t2) const

### 6.80.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo< _T1, _T2 >
```

Similar to `std::equal_to`, but allows two different types.

### 6.80.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [base.h](#)

## 6.81 std::\_\_detail::\_Executor< \_Bilter, \_Alloc, \_TraitsT > Class Template Reference

```
#include <regex_executor.h>
```

### Public Types

- typedef [iterator\\_traits](#)< \_Bilter >::value\_type \_CharT
- typedef \_TraitsT::char\_class\_type \_ClassT
- typedef [regex\\_constants::match\\_flag\\_type](#) \_FlagT
- typedef \_NFA< \_TraitsT > \_NFAT
- typedef [basic\\_regex](#)< \_CharT, \_TraitsT > \_RegexT
- typedef [::vector](#)< [sub\\_match](#)< \_Bilter >, \_Alloc > \_ResultsVec

### Public Member Functions

- **\_Executor** (\_Bilter \_\_begin, \_Bilter \_\_end, \_ResultsVec &\_\_results, const \_RegexT &\_\_re, [\\_FlagT](#) \_\_flags, bool \_\_use\_dfs)
- bool **\_M\_match** ()
- bool **\_M\_search** ()
- bool **\_M\_search\_from\_first** ()

### Public Attributes

- \_Bilter **\_M\_begin**
- \_ResultsVec **\_M\_cur\_results**
- \_Bilter **\_M\_current**
- const \_Bilter **\_M\_end**
- [\\_FlagT](#) **\_M\_flags**
- [::vector](#)< \_ExecutorFrame< \_Bilter > > **\_M\_frames**

- `bool _M_has_sol`
- `::vector< pair<_StateIdT, _ResultsVec>> _M_match_queue`
- `const _NFAT & _M_nfa`
- `const _RegexT & _M_re`
- `::vector< pair<_Bilter, int>> _M_rep_count`
- `_ResultsVec & _M_results`
- `const _Search_mode _M_search_mode`
- `_Bilter _M_sol_pos`
- `_StateIdT _M_start`
- `bool * _M_visited_states`

### 6.81.1 Detailed Description

`template<typename _Bilter, typename _Alloc, typename _TraitsT>`  
`class std::__detail::_Executor<_Bilter, _Alloc, _TraitsT>`

Takes a regex and an input string and does the matching.

The `_Executor` class has two modes: DFS mode and BFS mode, controlled by the function parameter `__search_mode`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex\\_executor.h](#)
- [regex\\_executor.tcc](#)

## 6.82 `__gnu_cxx::_ExtPtr_allocator<_Tp>` Class Template Reference

```
#include <extptr_allocator.h>
```

### Public Types

- `typedef _Pointer_adapter<_Relative_pointer_impl< const _Tp>> const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Pointer_adapter<_Relative_pointer_impl< _Tp>> pointer`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

### Public Member Functions

- `_ExtPtr_allocator (const _ExtPtr_allocator &__arg) noexcept`
- `template<typename _Up>`  
`_ExtPtr_allocator (const _ExtPtr_allocator< _Up> &__arg) noexcept`
- `const std::allocator< _Tp> & _M_getUnderlyingImp () const`
- `const_pointer address (const_reference __x) const noexcept`
- `pointer address (reference __x) const noexcept`
- `pointer allocate (size_type __n, const void *__p=0)`
- `template<typename _Up, typename... _Args>`  
`void construct (_Up *__p, _Args &&... __args)`
- `template<typename... _Args>`  
`void construct (pointer __p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Up>`  
`void destroy (_Up *__p)`

- void **destroy** (pointer \_\_p)
- size\_type **max\_size** () const noexcept
- bool **operator==** (const \_ExtPtr\_allocator &\_\_rarg) const
- template<typename \_Up>  
bool **operator==** (const \_ExtPtr\_allocator< \_Up > &\_\_rarg) const

### Friends

- template<typename \_Up>  
void **swap** (\_ExtPtr\_allocator< \_Up > &, \_ExtPtr\_allocator< \_Up > &)

#### 6.82.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator< _Tp >
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

#### 6.83 \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder< \_Tp > Class Template Reference

```
#include <bitmap_allocator.h>
```

### Public Types

- typedef \_Block\_pair **argument\_type**
- typedef bool **result\_type**

### Public Member Functions

- std::size\_t \* **\_M\_get** () const throw ()
- \_Counter\_type **\_M\_offset** () const throw ()
- bool **operator()** (\_Block\_pair \_\_bp) throw ()

#### 6.83.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder< _Tp >
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

#### 6.84 std::\_Function\_base Class Reference

```
#include <std_function.h>
```

Inheritance diagram for std::\_Function\_base:



#### Public Types

- using **\_Manager\_type**

#### Public Member Functions

- bool **\_M\_empty** () const

#### Public Attributes

- **\_Any\_data** **\_M\_functor**
- **\_Manager\_type** **\_M\_manager**

#### Static Public Attributes

- static const size\_t **\_M\_max\_align**
- static const size\_t **\_M\_max\_size**

#### 6.84.1 Detailed Description

Base class of all polymorphic function object wrappers.

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

## 6.85 std::\_Fwd\_list\_base< \_Tp, \_Alloc > Struct Template Reference

```
#include <forward_list.h>
```

Inheritance diagram for `std::_Fwd_list_base<_Tp, _Alloc>`:



### Public Types

- using **const\_iterator**
- using **iterator**
- using **pointer**

### Public Member Functions

- **\_Fwd\_list\_base** (`_Fwd_list_base &&`)=default
- **\_Fwd\_list\_base** (`_Fwd_list_base &&__lst, _Node_alloc_type &&__a`)
- **\_Fwd\_list\_base** (`_Fwd_list_base &&__lst, _Node_alloc_type &&__a, std::true\_type`)
- **\_Fwd\_list\_base** (`_Node_alloc_type &&__a`)
- `const _Node_alloc_type & _M_get_Node_allocator ()` `const noexcept`
- `_Node_alloc_type & _M_get_Node_allocator ()` `noexcept`

### Protected Types

- using **\_Base\_ptr**
- using **\_Node**
- using **\_Node\_alloc\_traits**
- using **\_Node\_alloc\_type**
- using **\_Node\_ptr**
- using **\_Node\_traits**

### Protected Member Functions

- `template<typename... _Args>`  
`_Node_ptr _M_create_node (_Args &&... __args)`
- `void _M_destroy_node (_Node_ptr __p)`
- `_Base_ptr _M_erase_after (_Base_ptr __pos)`
- `_Base_ptr _M_erase_after (_Base_ptr __pos, _Base_ptr __last)`
- `_Node * _M_get_node ()`
- `template<typename... _Args>`  
`_Base_ptr _M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void _M_put_node (_Node_ptr __p)`

**Protected Attributes**

- `_Fwd_list_impl _M_impl`

**6.85.1 Detailed Description**

```
template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >
```

Base class for `forward_list`.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

**6.86 std::\_Fwd\_list\_const\_iterator< \_Tp > Struct Template Reference**

```
#include <forward_list.h>
```

**Public Types**

- typedef const [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_const\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **iterator**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**

**Public Member Functions**

- **\_Fwd\_list\_const\_iterator** (const [\\_Fwd\\_list\\_node\\_base](#) \*\_\_n) noexcept
- **\_Fwd\_list\_const\_iterator** (const iterator &\_\_iter) noexcept
- reference **operator\*** () const noexcept
- **\_Self** & **operator++** () noexcept
- **\_Self** **operator++** (int) noexcept
- pointer **operator->** () const noexcept

**Friends**

- template<typename, typename>  
struct **\_Fwd\_list\_base**
- template<typename, typename>  
class **forward\_list**
- bool [operator==](#) (const \_Self &\_\_x, const \_Self &\_\_y) noexcept

**6.86.1 Detailed Description**

```
template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >
```

A `forward_list::const_iterator`.

All the functions are op overloads.

## 6.86.2 Friends And Related Symbol Documentation

### operator==

```
template<typename _Tp>
bool operator== (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list const\_iterator equality comparison.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 6.87 std::\_Fwd\_list\_iterator< \_Tp > Struct Template Reference

```
#include <forward_list.h>
```

### Public Types

- typedef [\\_Fwd\\_list\\_node](#)< \_Tp > **\_Node**
- typedef [\\_Fwd\\_list\\_iterator](#)< \_Tp > **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

### Public Member Functions

- [\\_Fwd\\_list\\_iterator](#) ([\\_Fwd\\_list\\_node\\_base](#) \*\_\_n) noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept

### Friends

- template<typename, typename>  
struct **\_Fwd\_list\_base**
- struct **\_Fwd\_list\_const\_iterator**< \_Tp >
- template<typename, typename>  
class **forward\_list**
- bool [operator==](#) (const [\\_Self](#) &\_\_x, const [\\_Self](#) &\_\_y) noexcept

### 6.87.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >
```

A forward\_list::iterator.

All the functions are op overloads.



### 6.87.2 Friends And Related Symbol Documentation

#### operator==

```
template<typename _Tp>
bool operator== (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list iterator equality comparison.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 6.88 std::\_Fwd\_list\_node< \_Tp > Struct Template Reference

```
#include <forward_list.h>
```

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



#### Public Types

- using **\_Base\_ptr**
- using **\_Node\_ptr**

#### Public Member Functions

- **\_Fwd\_list\_node\_base \* \_M\_base\_ptr ()**
- **const \_Fwd\_list\_node\_base \* \_M\_base\_ptr () const**
- **\_Node\_ptr \_M\_node\_ptr ()**
- **void \_M\_reverse\_after () noexcept**
- **\_Fwd\_list\_node\_base \* \_M\_transfer\_after (\_Fwd\_list\_node\_base \* \_\_begin, \_Fwd\_list\_node\_base \* \_\_end) noexcept**
- **const \_Tp \* \_M\_valptr () const noexcept**
- **\_Tp \* \_M\_valptr () noexcept**

#### Public Attributes

- **\_Fwd\_list\_node\_base \* \_M\_next**
- **\_\_gnu\_cxx::\_\_aligned\_buffer< \_Tp > \_M\_storage**

### 6.88.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_node< _Tp >
```

A helper node class for `forward_list`. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 6.89 std::\_Fwd\_list\_node\_base Struct Reference

```
#include <forward_list.h>
```

Inheritance diagram for `std::_Fwd_list_node_base`:



#### Public Types

- using `_Base_ptr`

#### Public Member Functions

- `_Fwd_list_node_base` (`_Fwd_list_node_base &&__x`) `noexcept`
- `_Fwd_list_node_base` (`const _Fwd_list_node_base &`) `=delete`
- `_Fwd_list_node_base * _M_base_ptr` ()
- `const _Fwd_list_node_base * _M_base_ptr` () `const`
- `void _M_reverse_after` () `noexcept`
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`, `_Fwd_list_node_base * __end`) `noexcept`
- `_Fwd_list_node_base & operator=` (`_Fwd_list_node_base &&__x`) `noexcept`
- `_Fwd_list_node_base & operator=` (`const _Fwd_list_node_base &`) `=delete`

#### Public Attributes

- `_Fwd_list_node_base * _M_next`

### 6.89.1 Detailed Description

A helper basic node class for `forward_list`.

```
This is just a linked list with nothing inside it.
There are purely list shuffling utility methods here.
```

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 6.90 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>` Class Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- [\\_GuardedIterator](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Compare](#) &\_\_comp)
- [operator \\_RAIter](#) () const
- [std::iterator\\_traits<\\_RAIter>::value\\_type & operator\\*](#) () const
- [\\_GuardedIterator<\\_RAIter, \\_Compare> & operator++](#) ()

### Friends

- bool [operator<](#) (const [\\_GuardedIterator<\\_RAIter, \\_Compare>](#) &\_\_bi1, const [\\_GuardedIterator<\\_RAIter, \\_Compare>](#) &\_\_bi2)
- bool [operator<=](#) (const [\\_GuardedIterator<\\_RAIter, \\_Compare>](#) &\_\_bi1, const [\\_GuardedIterator<\\_RAIter, \\_Compare>](#) &\_\_bi2)

### 6.90.1 Detailed Description

```
template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator<_RAIter, _Compare>
```

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons. The implicit supremum comes with a performance cost. Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

### 6.90.2 Constructor & Destructor Documentation

#### `_GuardedIterator()`

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator<_RAIter, _Compare>::_GuardedIterator (
 _RAIter __begin,
 _RAIter __end,
 _Compare & __comp) [inline]
```

Constructor. Sets iterator to beginning of sequence.

### Parameters

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| <code>__begin</code> | Begin iterator of sequence.                                      |
| <code>__end</code>   | End iterator of sequence.                                        |
| <code>__comp</code>  | Comparator provided for associated overloaded compare operators. |

Referenced by [operator<](#), and [operator<=](#).

### 6.90.3 Member Function Documentation

#### **operator\_RAlter()**

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () const [inline]
```

Convert to wrapped iterator.

##### Returns

Wrapped iterator.

#### **operator\*()**

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::value_type & __gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator* () const [inline]
```

Dereference operator.

##### Returns

Referenced element.

#### **operator++()**

```
template<typename _RAIter, typename _Compare>
__GuardedIterator< _RAIter, _Compare > & __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::←
operator++ () [inline]
```

Pre-increment operator.

##### Returns

This.

### 6.90.4 Friends And Related Symbol Documentation

#### **operator<**

```
template<typename _RAIter, typename _Compare>
bool operator< (
 const __GuardedIterator< _RAIter, _Compare > & __bi1,
 const __GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

##### Parameters

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

##### Returns

`true` if less.

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

**operator<=**

```
template<typename _RAIter, typename _Compare>
bool operator<= (
 const _GuardedIterator< _RAIter, _Compare > & __bi1,
 const _GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

**Parameters**

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

**Returns**

True if less equal.

References `_GuardedIterator()`.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

**6.91 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>` Struct Template Reference**

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`:

**Public Types**

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits>::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**

- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `base_type::type_traits` **type\_traits**

#### Public Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_min**
- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

#### 6.91.1 Detailed Description

**template**<typename `_ATraits`, typename `Metadata`>  
**struct** `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`

Head node for PATRICIA tree.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 6.92 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>` Struct Template Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`:



#### Classes

- struct [const\\_iterator](#)
- struct [iterator](#)

## Public Types

- enum { **arr\_size** }
- typedef `detail::rebind_traits<_Alloc, node_pointer>` **\_\_rebind\_np**
- typedef `base_type::allocator_type` **\_Alloc**
- typedef `base_type::access_traits` **access\_traits**
- typedef `_Alloc allocator_type`
- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `__rebind_np::pointer` **node\_pointer\_pointer**
- typedef `__rebind_np::reference` **node\_pointer\_reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `base_type::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

## Public Member Functions

- **\_Inode** (size\_type, const a\_const\_iterator)
- node\_pointer **add\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- iterator **begin** ()
- const\_iterator **begin** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **get\_child\_it** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_const\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer) const
- size\_type **get\_e\_ind** () const
- node\_const\_pointer **get\_join\_child** (node\_const\_pointer, a\_const\_pointer) const
- node\_pointer **get\_join\_child** (node\_pointer, a\_const\_pointer)
- node\_pointer **get\_lower\_bound\_child\_node** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer)
- leaf\_pointer **leftmost\_descendant** ()
- leaf\_const\_pointer **leftmost\_descendant** () const
- a\_const\_iterator **pref\_b\_it** () const
- a\_const\_iterator **pref\_e\_it** () const
- void **remove\_child** (iterator)
- void **remove\_child** (node\_pointer)
- void **replace\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- leaf\_pointer **rightmost\_descendant** ()
- leaf\_const\_pointer **rightmost\_descendant** () const
- bool **should\_be\_mine** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer) const
- void **update\_prefixes** (a\_const\_pointer)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const `node_type` **m\_type**

### 6.92.1 Detailed Description

template<typename **\_ATraits**, typename **Metadata**>  
**struct** `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`

Internal node type, PATRICIA tree.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.93 `__gnu_cxx::_Invalid_type` Struct Reference

```
#include <pointer.h>
```

### 6.93.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 6.94 `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



### Public Types

- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional\_iterator\_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`



**Public Member Functions**

- **\_Iter** (const \_Iter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- **\_Iter** (node\_pointer p\_nd=0)
- bool **operator!=** (const \_CIter &other) const
- bool **operator!=** (const \_CIter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- reference **operator\*** () const
- **\_Iter & operator++** ()
- **\_Iter operator++** (int)
- **\_Iter & operator--** ()
- **\_Iter operator--** (int)
- pointer **operator->** () const
- **\_Iter & operator=** (const \_Iter &other)
- **\_Iter & operator=** (const \_Iter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool **operator==** (const \_CIter &other) const
- bool **operator==** (const \_CIter< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const

**Public Attributes**

- node\_pointer **m\_p\_nd**

**Protected Member Functions**

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

**Static Protected Member Functions**

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

**6.94.1 Detailed Description**

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

**6.95 std::\_\_fwlist::\_Iterator<\_Const,\_Ptr> Class Template Reference**

```
#include <forward_list.h>
```

**Public Types**

- using **difference\_type**
- using **iterator\_category**
- using **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- constexpr **\_Iterator** (\_Base\_ptr \_\_x) noexcept
- **\_Iterator** (const \_Iterator &)=default
- constexpr **\_Iterator** (const \_Iterator< false, \_Ptr > &\_\_i)
- constexpr reference **operator\*** () const noexcept
- constexpr \_Iterator & **operator++** () noexcept
- constexpr \_Iterator **operator++** (int) noexcept
- constexpr pointer **operator->** () const noexcept
- \_Iterator & **operator=** (const \_Iterator &)=default

### Friends

- template<typename \_Tp, typename \_Allocator>  
struct **::\_Fwd\_list\_base**
- template<typename \_Tp, typename \_Allocator>  
class **::forward\_list**
- constexpr bool **operator==** (const \_Iterator &\_\_x, const \_Iterator &\_\_y) noexcept

#### 6.95.1 Detailed Description

```
template<bool _Const, typename _Ptr>
class std::__fwdlist::_Iterator< _Const, _Ptr >
```

A forward\_list iterator when the allocator uses fancy pointers.

#### 6.95.2 Friends And Related Symbol Documentation

##### **operator==**

```
template<bool _Const, typename _Ptr>
bool operator== (
 const _Iterator< _Const, _Ptr > & __x,
 const _Iterator< _Const, _Ptr > & __y) [friend]
```

Forward list iterator equality comparison.

The documentation for this class was generated from the following file:

- [forward\\_list.h](#)

#### 6.96 **\_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory > Class** Template Reference

```
#include <iterator.h>
```

Inheritance diagram for `__gnu_parallel::IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>`:



### Public Types

- typedef `std::iterator_traits<_Iterator1>` `_TraitsType`
- typedef `_TraitsType::difference_type` `difference_type`
- typedef `_Iterator1` `first_type`
- typedef `_IteratorCategory` `iterator_category`
- typedef `_IteratorPair *` `pointer`
- typedef `_IteratorPair &` `reference`
- typedef `_Iterator2` `second_type`
- typedef void `value_type`

### Public Member Functions

- `_IteratorPair` (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second)
- constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) `pair`(`pair`<\_U1
- constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) `pair`(`pair`<\_U1
- constexpr `_U2` &&constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) `pair`(`pair`<\_U1
- constexpr `_U2` &&constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) `pair`(`pair`<\_U1
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`_U1` &&
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`_U1` &&
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`_U1` &&\_\_x
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`_U1` &&\_\_x
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`pair`<\_U1
- constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`pair`<\_U1
- constexpr `_U2` &constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`pair`<\_U1
- constexpr `_U2` &constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) `pair`(`pair`<\_U1
- constexpr **explicit** (!\_S\_convertible<const `_Iterator1` &, const `_Iterator2` &>()) `pair`(const `type_identity_t`<\_↵  
\_Iterator1 > &\_\_x
- constexpr **explicit** (!\_S\_convertible<const `_Iterator1` &, const `_Iterator2` &>()) `pair`(const `type_identity_t`<\_↵  
\_Iterator1 > &\_\_x
- constexpr **explicit** (!\_S\_convertible<const `_U1` &, const `_U2` &>()) `pair`(const `pair`<\_U1
- constexpr **explicit** (!\_S\_convertible<const `_U1` &, const `_U2` &>()) `pair`(const `pair`<\_U1

- constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible< const \_U1 &, const \_U2 &>()) pair(const pair< \_U1
- constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible< const \_U1 &, const \_U2 &>()) pair(const pair< \_U1
- constexpr **explicit** (!\_S\_convertible< const \_U1, const \_U2 >()) pair(const pair< \_U1
- constexpr **explicit** (!\_S\_convertible< const \_U1, const \_U2 >()) pair(const pair< \_U1
- constexpr \_U2 &constexpr **explicit** (!\_S\_convertible< const \_U1, const \_U2 >()) pair(const pair< \_U1
- constexpr \_U2 &constexpr **explicit** (!\_S\_convertible< const \_U1, const \_U2 >()) pair(const pair< \_U1
- constexpr **explicit** (!\_S\_convertible\_from\_pair\_like< \_UPair >()) pair(\_UPair &&)=delete
- constexpr **explicit** (!\_S\_convertible\_from\_pair\_like< \_UPair >()) pair(\_UPair &&)=delete
- constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible\_from\_pair\_like< \_UPair >()) pair(\_UPair &&\_p)
- constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible\_from\_pair\_like< \_UPair >()) pair(\_UPair &&\_p)
- constexpr **explicit** (\_\_not\_\_< \_\_and\_\_< \_\_is\_implicitly\_default\_constructible< \_Iterator1 >, \_\_is\_implicitly\_↵  
default\_constructible< \_Iterator2 > >>()) pair() noexcept(is\_nothrow\_default\_constructible\_v< \_Iterator1 >  
&&is\_nothrow\_default\_constructible\_v< \_Iterator2 >)
- constexpr **explicit** (\_\_not\_\_< \_\_and\_\_< \_\_is\_implicitly\_default\_constructible< \_Iterator1 >, \_\_is\_implicitly\_↵  
default\_constructible< \_Iterator2 > >>()) pair() noexcept(is\_nothrow\_default\_constructible\_v< \_Iterator1 >  
&&is\_nothrow\_default\_constructible\_v< \_Iterator2 >)
- constexpr \_U2 &&constexpr \_U2 &\_p **noexcept** (\_S\_nothrow\_constructible< \_U1 &, \_U2 &>())
- constexpr \_U2 &&constexpr \_U2 &\_p **noexcept** (\_S\_nothrow\_constructible< \_U1 &, \_U2 &>())
- constexpr \_U2 &constexpr \_U2 &&\_p **noexcept** (\_S\_nothrow\_constructible< \_U1, \_U2 >())
- constexpr \_U2 &constexpr \_U2 &&\_p **noexcept** (\_S\_nothrow\_constructible< \_U1, \_U2 >())
- constexpr \_U2 &&\_y **noexcept** (\_S\_nothrow\_constructible< \_U1, \_U2 >())
- constexpr \_U2 &&\_y **noexcept** (\_S\_nothrow\_constructible< \_U1, \_U2 >())
- constexpr const \_Iterator2 &\_y **noexcept** (\_S\_nothrow\_constructible< const \_Iterator1 &, const \_Iterator2 &  
>())
- constexpr const \_Iterator2 &\_y **noexcept** (\_S\_nothrow\_constructible< const \_Iterator1 &, const \_Iterator2 &  
>())
- constexpr \_U2 &&constexpr \_U2 &\_p **noexcept** (\_S\_nothrow\_constructible< const \_U1 &, const \_U2 &>())
- constexpr \_U2 &&constexpr \_U2 &\_p **noexcept** (\_S\_nothrow\_constructible< const \_U1 &, const \_U2 &>())
- constexpr \_U2 &constexpr \_U2 &&\_p **noexcept** (\_S\_nothrow\_constructible< const \_U1, const \_U2 >())
- constexpr \_U2 &constexpr \_U2 &&\_p **noexcept** (\_S\_nothrow\_constructible< const \_U1, const \_U2 >())
- **operator** \_Iterator2 () const
- \_IteratorPair **operator**+ (difference\_type \_\_delta) const
- \_IteratorPair & **operator**++ ()
- const \_IteratorPair **operator**++ (int)
- difference\_type **operator**- (const \_IteratorPair &\_\_other) const
- \_IteratorPair & **operator**-- ()
- const \_IteratorPair **operator**-- (int)
- \_IteratorPair & **operator**= (const \_IteratorPair &\_\_other)
- constexpr void **swap** (const pair &\_\_p) const noexcept(\_\_and\_v< \_\_is\_nothrow\_swappable< const \_Iterator1 >,  
\_\_is\_nothrow\_swappable< const \_Iterator2 > >)
- constexpr void **swap** (const pair &\_\_p) const noexcept(\_\_and\_v< \_\_is\_nothrow\_swappable< const \_Iterator1 >,  
\_\_is\_nothrow\_swappable< const \_Iterator2 > >)
- constexpr void **swap** (pair &\_\_p) noexcept(\_\_and\_\_< \_\_is\_nothrow\_swappable< \_Iterator1 >, \_\_is\_nothrow\_↵  
swappable< \_Iterator2 > >::value)
- constexpr void **swap** (pair &\_\_p) noexcept(\_\_and\_\_< \_\_is\_nothrow\_swappable< \_Iterator1 >, \_\_is\_nothrow\_↵  
swappable< \_Iterator2 > >::value)

## Public Attributes

- \_Iterator1 **first**
- \_Iterator1 **first**
- \_Iterator2 **second**
- \_Iterator2 **second**

## Related Symbols

(Note that these are not member symbols.)

- constexpr enable\_if< \_\_and< \_\_is\_swappable< \_Iterator1 >, \_\_is\_swappable< \_Iterator2 > >::value >::type `swap` (`pair`< \_Iterator1, \_Iterator2 > &\_\_x, `pair`< \_Iterator1, \_Iterator2 > &\_\_y) noexcept(noexcept(\_\_x.swap(↵\_\_y)))
- constexpr enable\_if< \_\_and< \_\_is\_swappable< \_Iterator1 >, \_\_is\_swappable< \_Iterator2 > >::value >::type `swap` (`pair`< \_Iterator1, \_Iterator2 > &\_\_x, `pair`< \_Iterator1, \_Iterator2 > &\_\_y) noexcept(noexcept(\_\_x.swap(↵\_\_y)))
- constexpr bool `operator==` (const `pair`< \_Iterator1, \_Iterator2 > &\_\_x, const `pair`< \_U1, \_U2 > &\_\_y)
- constexpr bool `operator==` (const `pair`< \_Iterator1, \_Iterator2 > &\_\_x, const `pair`< \_U1, \_U2 > &\_\_y)
- constexpr common\_comparison\_category\_t< \_\_detail::\_\_synth3way\_t< \_Iterator1, \_U1 >, \_\_detail::\_\_synth3way\_t< \_Iterator2, \_U2 > > `operator<=` (const `pair`< \_Iterator1, \_Iterator2 > &\_\_x, const `pair`< \_U1, \_U2 > &\_\_y)
- constexpr common\_comparison\_category\_t< \_\_detail::\_\_synth3way\_t< \_Iterator1, \_U1 >, \_\_detail::\_\_synth3way\_t< \_Iterator2, \_U2 > > `operator<=` (const `pair`< \_Iterator1, \_Iterator2 > &\_\_x, const `pair`< \_U1, \_U2 > &\_\_y)

### 6.96.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

### 6.96.2 Member Typedef Documentation

#### first\_type

```
typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type [inherited]
```

The type of the first member.

#### second\_type

```
typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
```

The type of the second member.

### 6.96.3 Member Function Documentation

#### explicit() [1/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1 &, _U2 & > ()) [constexpr], [inherited]
```

Converting constructor from a non-const pair<U1, U2> lvalue.

#### explicit() [2/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1 &, _U2 & > ()) [constexpr], [inherited]
```

Converting constructor from a non-const pair<U1, U2> lvalue.

#### explicit() [3/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) && [constexpr], [inherited]
```

Constructor accepting two values of arbitrary types.

**explicit()** [4/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) && [constexpr], [inherited]
```

Constructor accepting two values of arbitrary types.

**explicit()** [5/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) [constexpr], [inherited]
```

Converting constructor from a non-const pair<U1, U2> rvalue.

**explicit()** [6/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) [constexpr], [inherited]
```

Converting constructor from a non-const pair<U1, U2> rvalue.

**explicit()** [7/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _Iterator1 &, const _Iterator2 & > ()) const & [constexpr],
[inherited]
```

Constructor accepting lvalues of first\_type and second\_type.

**explicit()** [8/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _Iterator1 &, const _Iterator2 & > ()) const & [constexpr],
[inherited]
```

Constructor accepting lvalues of first\_type and second\_type.

**explicit()** [9/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _U1 &, const _U2 & > ()) const [constexpr], [inherited]
```

Converting constructor from a const pair<U1, U2> lvalue.

**explicit()** [10/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _U1 &, const _U2 & > ()) const [constexpr], [inherited]
```

Converting constructor from a const pair<U1, U2> lvalue.

**explicit()** [11/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _U1, const _U2 > ()) const [constexpr], [inherited]
```

Converting constructor from a const pair<U1, U2> rvalue.

**explicit()** [12/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
 !_S_convertible< const _U1, const _U2 > ()) const [constexpr], [inherited]
```

Converting constructor from a const pair<U1, U2> rvalue.

**explicit()** [13/14]

```
std::pair<_Iterator1, _Iterator2>::explicit (
 __not_< __and_< __is_implicitly_default_constructible<_Iterator1>, __is_implicitly_↵
 _default_constructible<_Iterator2>>> () [inline], [constexpr], [noexcept], [inherited]
```

Default constructor.

**explicit()** [14/14]

```
std::pair<_Iterator1, _Iterator2>::explicit (
 __not_< __and_< __is_implicitly_default_constructible<_Iterator1>, __is_implicitly_↵
 _default_constructible<_Iterator2>>> () [inline], [constexpr], [noexcept], [inherited]
```

Default constructor.

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

```
void std::pair<_Iterator1, _Iterator2>::swap (
 pair<_Iterator1, _Iterator2> & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

**swap()** [2/2]

```
void std::pair<_Iterator1, _Iterator2>::swap (
 pair<_Iterator1, _Iterator2> & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

**6.96.4 Friends And Related Symbol Documentation****operator<=>()** [1/2]

```
common_comparison_category_t< __detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_↵
_t<_Iterator2, _U2>> operator<=> (
 const pair<_Iterator1, _Iterator2> & __x,
 const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

**operator<=>()** [2/2]

```
common_comparison_category_t< __detail::__synth3way_t<_Iterator1, _U1>, __detail::__synth3way_↵
_t<_Iterator2, _U2>> operator<=> (
 const pair<_Iterator1, _Iterator2> & __x,
 const pair<_U1, _U2> & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

**operator==()** [1/2]

```
bool operator== (
 const pair<_Iterator1, _Iterator2> & __x,
 const pair<_U1, _U2> & __y) [related]
```

Two pairs are equal iff their members are equal.

**operator==( )** [2/2]

```
bool operator== (
 const pair< _Iterator1, _Iterator2 > & __x,
 const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

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

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
 pair< _Iterator1, _Iterator2 > & __x,
 pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

**swap( )** [2/2]

```
enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
 pair< _Iterator1, _Iterator2 > & __x,
 pair< _Iterator1, _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

**Note**

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

## 6.96.5 Member Data Documentation

**first** [1/2]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first [inherited]
The first member.
```

**first** [2/2]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first [inherited]
The first member.
```

**second** [1/2]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second [inherited]
The second member.
```

**second** [2/2]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second [inherited]
The second member.
```

The documentation for this class was generated from the following file:

- [iterator.h](#)



## 6.97 `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

```
#include <iterator.h>
```

### Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` **difference\_type**
- typedef `_IteratorCategory` **iterator\_category**
- typedef `_IteratorTriple *` **pointer**
- typedef `_IteratorTriple &` **reference**
- typedef void **value\_type**

### Public Member Functions

- **\_IteratorTriple** (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second, const `_Iterator3` &\_\_third)
- **operator \_Iterator3** () const
- `_IteratorTriple` **operator+** (difference\_type \_\_delta) const
- `_IteratorTriple` & **operator++** ()
- const `_IteratorTriple` **operator++** (int)
- difference\_type **operator-** (const `_IteratorTriple` &\_\_other) const
- `_IteratorTriple` & **operator--** ()
- const `_IteratorTriple` **operator--** (int)
- `_IteratorTriple` & **operator=** (const `_IteratorTriple` &\_\_other)

### Public Attributes

- `_Iterator1` **\_M\_first**
- `_Iterator2` **\_M\_second**
- `_Iterator3` **\_M\_third**

#### 6.97.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>
class __gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.  
The documentation for this class was generated from the following file:

- [iterator.h](#)

## 6.98 `__gnu_parallel::Job<_DifferenceTp>` Struct Template Reference

```
#include <workstealing.h>
```

### Public Types

- typedef `_DifferenceTp` **\_DifferenceType**

### Public Attributes

- volatile `_DifferenceType` **\_M\_first**
- volatile `_DifferenceType` **\_M\_last**
- volatile `_DifferenceType` **\_M\_load**

### 6.98.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Job< _DifferenceTp >
```

One \_\_job for a certain thread.

### 6.98.2 Member Data Documentation

#### **\_\_M\_first**

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::__M_first
```

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

#### **\_\_M\_last**

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::__M_last
```

Last element.

Changed by owning thread only.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

#### **\_\_M\_load**

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::__M_load
```

Number of elements, i.e. `__M_last-__M_first+1`.

Changed by owning thread only.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing\(\)](#).

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

## 6.99 `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Leaf<\_ATraits, Metadata >:



## Public Types

- typedef \_ATraits::const\_iterator **a\_const\_iterator**
- typedef [detail::rebind\\_traits](#)<\_Alloc, \_ATraits >::const\_pointer **a\_const\_pointer**
- typedef \_ATraits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef \_Node\_base<\_ATraits, Metadata > **base\_type**
- typedef type\_traits::const\_reference **const\_reference**
- typedef [detail::rebind\\_traits](#)<\_Alloc, \_Node\_base >::pointer **node\_pointer**
- typedef type\_traits::reference **reference**
- typedef base\_type::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

## Public Member Functions

- **\_Leaf** (const\_reference other)
- reference **value** ()
- const\_reference **value** () const

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

### 6.99.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata >
```

Leaf node for PATRICIA tree.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.100 \_\_gnu\_parallel::\_Less< \_T1, \_T2 > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_Less< \_T1, \_T2 >:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

#### 6.100.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_Less< _T1, _T2 >
```

Similar to `std::less`, but allows two different types.

#### 6.100.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [base.h](#)

**6.101 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>` Class Template Reference**

```
#include <multiseq_selection.h>
```

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:

**Public Types**

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

**Public Member Functions**

- `_Lexicographic` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2> &__p1, const std::pair<_T1, _T2> &__p2`) `const`

**6.101.1 Detailed Description**

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare>
```

Compare `__a` pair of types lexicographically, ascending.

**6.101.2 Member Typedef Documentation****first\_argument\_type**

```
typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>,
bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

### result\_type

```
typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::↵
result_type [inherited]
result_type is the return type
```

### second\_argument\_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2
>, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:
```

- [multiseq\\_selection.h](#)

## 6.102 `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare >` Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare >`:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- bool `operator()` (const [std::pair](#)< `_T1`, `_T2` > &`__p1`, const [std::pair](#)< `_T1`, `_T2` > &`__p2`) const

### 6.102.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>
```

Compare \_\_a pair of types lexicographically, descending.

### 6.102.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function<_T1, _T2, bool>::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

## 6.103 std::\_List\_base<\_Tp, \_Alloc> Class Template Reference

```
#include <stl_list.h>
```

Inheritance diagram for std::\_List\_base<\_Tp, \_Alloc>:



### Public Types

- typedef \_Alloc **allocator\_type**

## Public Member Functions

- **\_List\_base** (\_List\_base &&)=default
- **\_List\_base** (\_List\_base && \_\_x, \_Node\_alloc\_type && \_\_a)
- **\_List\_base** (\_Node\_alloc\_type && \_\_a)
- **\_List\_base** (\_Node\_alloc\_type && \_\_a, \_List\_base && \_\_x)
- **\_List\_base** (const \_Node\_alloc\_type & \_\_a) noexcept
- void **\_M\_clear** () noexcept
- size\_t **\_M\_distance** (const [\\_\\_detail::List\\_node\\_base](#) \* \_\_first, const [\\_\\_detail::List\\_node\\_base](#) \* \_\_last) const
- const \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () const noexcept
- \_Node\_alloc\_type & **\_M\_get\_Node\_allocator** () noexcept
- void **\_M\_init** () noexcept
- void **\_M\_move\_nodes** (\_List\_base && \_\_x)
- size\_t **\_M\_node\_count** () const

## Static Public Member Functions

- static size\_t **\_S\_distance** (const [\\_\\_detail::List\\_node\\_base](#) \* \_\_first, const [\\_\\_detail::List\\_node\\_base](#) \* \_\_last)

## Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Node\_alloc\_type > **\_Node\_alloc\_traits**
- typedef \_Tp\_alloc\_traits::template rebind< typename \_Node\_traits::\_Node >::other **\_Node\_alloc\_type**
- using **\_Node\_ptr**
- typedef \_\_list:: \_Node\_traits< \_Tp, typename [\\_Tp\\_alloc\\_traits::pointer](#) > **\_Node\_traits**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Tp\_alloc\_type > **\_Tp\_alloc\_traits**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Tp >::other **\_Tp\_alloc\_type**

## Protected Member Functions

- void **\_M\_dec\_size** (size\_t \_\_n)
- void **\_M\_destroy\_node** (\_Node\_ptr \_\_p)
- [\\_Node\\_alloc\\_traits::pointer](#) **\_M\_get\_node** ()
- size\_t **\_M\_get\_size** () const
- void **\_M\_inc\_size** (size\_t \_\_n)
- void **\_M\_put\_node** (\_Node\_ptr \_\_p) noexcept
- void **\_M\_set\_size** (size\_t \_\_n)

## Protected Attributes

- **\_List\_impl** **\_M\_impl**

### 6.103.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >
```

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)



## 6.104 `std::_List_const_iterator< _Tp >` Struct Template Reference

```
#include <stl_list.h>
```

### Public Types

- `typedef const _List_node< _Tp > _Node`
- `typedef ptrdiff_t difference_type`
- `typedef _List_iterator< _Tp > iterator`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

### Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base * __x`) noexcept
- `_List_const_iterator` (const iterator & \_\_x) noexcept
- `iterator _M_const_cast ()` const noexcept
- `reference operator* ()` const noexcept
- `_List_const_iterator & operator++ ()` noexcept
- `_List_const_iterator operator++ (int)` noexcept
- `_List_const_iterator & operator-- ()` noexcept
- `_List_const_iterator operator-- (int)` noexcept
- `pointer operator-> ()` const noexcept

### Public Attributes

- const `__detail::_List_node_base * _M_node`

### Friends

- `bool operator==` (const `_List_const_iterator` & \_\_x, const `_List_const_iterator` & \_\_y) noexcept

#### 6.104.1 Detailed Description

```
template<typename _Tp>
struct std::_List_const_iterator< _Tp >
```

A `list::const_iterator`.

All the functions are op overloads.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

## 6.105 `std::_List_iterator< _Tp >` Struct Template Reference

```
#include <stl_list.h>
```

### Public Types

- typedef [\\_List\\_node](#)<\_Tp> **\_Node**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

### Public Member Functions

- **\_List\_iterator** ([\\_\\_detail::\\_List\\_node\\_base](#) \*\_\_x) noexcept
- **\_List\_iterator** **\_M\_const\_cast** () const noexcept
- reference **operator\*** () const noexcept
- **\_List\_iterator** & **operator++** () noexcept
- **\_List\_iterator** **operator++** (int) noexcept
- **\_List\_iterator** & **operator--** () noexcept
- **\_List\_iterator** **operator--** (int) noexcept
- pointer **operator->** () const noexcept

### Public Attributes

- [\\_\\_detail::\\_List\\_node\\_base](#) \* **\_M\_node**

### Friends

- bool **operator==** (const **\_List\_iterator** &\_\_x, const **\_List\_iterator** &\_\_y) noexcept

#### 6.105.1 Detailed Description

```
template<typename _Tp>
struct std::_List_iterator<_Tp>
```

A list::iterator.

All the functions are op overloads.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

#### 6.106 **std::\_List\_node**<\_Tp> Struct Template Reference

```
#include <stl_list.h>
```

Inheritance diagram for `std::_List_node<_Tp>`:



### Public Types

- typedef `_List_node_base * _Base_ptr`
- typedef `_List_node * _Node_ptr`

### Public Member Functions

- `_List_node_base * _M_base ()`
- `const _List_node_base * _M_base () const`
- `void _M_hook (_List_node_base *const __position) noexcept`
- `_Node_ptr _M_node_ptr ()`
- `void _M_reverse () noexcept`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) noexcept`
- `void _M_unhook () noexcept`
- `_Tp * _M_valptr ()`
- `_Tp const * _M_valptr () const`

### Static Public Member Functions

- static void `swap (_List_node_base &__x, _List_node_base &__y) noexcept`

### Public Attributes

- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`
- `__gnu_cxx::__aligned_membuf<_Tp> _M_storage`

#### 6.106.1 Detailed Description

**template<typename \_Tp>**  
**struct** `std::_List_node<_Tp>`

An actual node in the list.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 6.107 std::\_\_detail::\_List\_node\_base Struct Reference

```
#include <stl_list.h>
```

Inheritance diagram for std::\_\_detail::\_List\_node\_base:



### Public Types

- typedef [\\_List\\_node\\_base](#) \* [\\_Base\\_ptr](#)

### Public Member Functions

- [\\_List\\_node\\_base](#) \* [\\_M\\_base](#) ()
- const [\\_List\\_node\\_base](#) \* [\\_M\\_base](#) () const
- void [\\_M\\_hook](#) ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void [\\_M\\_reverse](#) () noexcept
- void [\\_M\\_transfer](#) ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void [\\_M\\_unhook](#) () noexcept

### Static Public Member Functions

- static void [swap](#) ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

### Public Attributes

- [\\_List\\_node\\_base](#) \* [\\_M\\_next](#)
- [\\_List\\_node\\_base](#) \* [\\_M\\_prev](#)

#### 6.107.1 Detailed Description

Common part of a node in the list.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 6.108 std::\_\_detail::\_List\_node\_header Struct Reference

```
#include <stl_list.h>
```

Inheritance diagram for std::\_\_detail::\_List\_node\_header:



### Public Types

- typedef [\\_List\\_node\\_base](#) \* [\\_Base\\_ptr](#)

### Public Member Functions

- [\\_List\\_node\\_header](#) ([\\_List\\_node\\_header](#) &&\_\_x) noexcept
- [\\_List\\_node\\_base](#) \* [\\_M\\_base](#) ()
- const [\\_List\\_node\\_base](#) \* [\\_M\\_base](#) () const
- void [\\_M\\_hook](#) ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void [\\_M\\_init](#) () noexcept
- void [\\_M\\_move\\_nodes](#) ([\\_List\\_node\\_header](#) &&\_\_x)
- void [\\_M\\_reverse](#) () noexcept
- void [\\_M\\_transfer](#) ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void [\\_M\\_unhook](#) () noexcept

### Static Public Member Functions

- static void [swap](#) ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

### Public Attributes

- [\\_List\\_node\\_base](#) \* [\\_M\\_next](#)
- [\\_List\\_node\\_base](#) \* [\\_M\\_prev](#)
- [size\\_t](#) [\\_M\\_size](#)

#### 6.108.1 Detailed Description

The list node header.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 6.109 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

```
#include <losertree.h>
```

### Public Attributes

- `_Tp _M_key`
- `int _M_source`
- `bool _M_sup`

#### 6.109.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
```

Internal representation of a `_LoserTree` element.

#### 6.109.2 Member Data Documentation

##### `_M_key`

```
template<typename _Tp, typename _Compare>
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key
_M_key of the element in the _LoserTree.
```

##### `_M_source`

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
__index of the __source __sequence.
```

##### `_M_sup`

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup
flag, true iff this is a "maximum" __sentinel.
```

The documentation for this struct was generated from the following file:

- [losertree.h](#)

## 6.110 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference

```
#include <losertree.h>
```

### Public Attributes

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

#### 6.110.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
```

Internal representation of `_LoserTree` \_\_elements.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

## 6.111 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_log_k`
- unsigned int `_M_offset`

#### 6.111.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
```

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

#### 6.111.2 Member Function Documentation

##### `__delete_min_insert()`

```
template<bool __stable, typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
```

```

 _Tp __key,
 bool __sup) [inline]

```

Delete the smallest element and insert a new element from the previously smallest element's sequence.  
This implementation is stable.

### **\_\_get\_min\_source()**

```

template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline], [inherited]

```

#### **Returns**

the index of the sequence with the smallest element.

References [\\_M\\_losers](#).

### **\_\_insert\_start()**

```

template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline], [inherited]

```

Initializes the sequence "[\\_M\\_source](#)" with the element "[\\_\\_key](#)".

#### **Parameters**

|                          |                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------|
| <a href="#">__key</a>    | the element to insert                                                                        |
| <a href="#">__source</a> | <a href="#">__index</a> of the <a href="#">__source</a> <a href="#">__sequence</a>           |
| <a href="#">__sup</a>    | flag that determines whether the value to insert is an explicit <a href="#">__supremum</a> . |

References [\\_M\\_first\\_insert](#), and [\\_M\\_losers](#).

### **6.111.3 Member Data Documentation**

#### **[\\_M\\_log\\_k](#)**

```

template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]
log_2(_M_k)

```

Referenced by [\\_LoserTreeBase\(\)](#).

The documentation for this class was generated from the following file:

- [losertree.h](#)

### **6.112 [\\_\\_gnu\\_parallel::\\_LoserTree< false, \\_Tp, \\_Compare >](#) Class Template Reference**

```
#include <losertree.h>
```



Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_log_k`
- unsigned int `_M_offset`

#### 6.112.1 Detailed Description

template<typename `_Tp`, typename `_Compare`>  
class `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

#### 6.112.2 Member Function Documentation

##### `__delete_min_insert()`

```

template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
 _Tp __key,
 bool __sup) [inline]

```

Delete the `_M_ik` smallest element and insert the element `__key` instead.

**Parameters**

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__key</code> | the <code>_M_key</code> to insert                            |
| <code>__sup</code> | true iff <code>__key</code> is an explicitly marked supremum |

**`__get_min_source()`**

```
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline]
```

**Returns**

the index of the sequence with the smallest element.

**`__init_winner()`**

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner (
 unsigned int __root) [inline]
```

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__root</code> | <code>__index</code> of the "game" to start. |
|---------------------|----------------------------------------------|

References [\\_\\_init\\_winner\(\)](#).

Referenced by [\\_\\_init\\_winner\(\)](#).

**`__insert_start()`**

```
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

**Parameters**

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                                     |
| <code>__source</code> | <code>__index</code> of the <code>__source</code> <code>__sequence</code>                 |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |

**6.112.3 Member Data Documentation****`_M_log_k`**

```
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
```

```
log_2[_M_k]
```

The documentation for this class was generated from the following file:

- [losertree.h](#)

**6.113 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>` Class Template Reference**

```
#include <losetree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>`:

**Classes**

- struct [\\_Loser](#)

**Public Member Functions**

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~\\_LoserTreeBase](#) ()
- int [\\_\\_get\\_min\\_source](#) ()
- void [\\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

**Protected Attributes**

- \_Compare [\\_M\\_comp](#)
- bool [\\_M\\_first\\_insert](#)
- unsigned int [\\_M\\_ik](#)
- unsigned int [\\_M\\_k](#)
- unsigned int [\\_M\\_log\\_k](#)
- [\\_Loser](#) \* [\\_M\\_losers](#)
- unsigned int [\\_M\\_offset](#)

**6.113.1 Detailed Description**

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase<_Tp, _Compare>
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine.

This is a well-performing variant.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>_Tp</code> | the element type |
|------------------|------------------|

|                       |                                                                      |
|-----------------------|----------------------------------------------------------------------|
| <code>_Compare</code> | the comparator to use, defaults to <code>std::less&lt;_Tp&gt;</code> |
|-----------------------|----------------------------------------------------------------------|

### 6.113.2 Constructor & Destructor Documentation

#### `_LoserTreeBase()`

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
 unsigned int __k,
 _Compare __comp) [inline]
```

The constructor.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__k</code>    | The number of sequences to merge. |
| <code>__comp</code> | The comparator to use.            |

References [\\_\\_gnu\\_parallel::\\_rd\\_log2\(\)](#), [\\_M\\_comp](#), [\\_M\\_first\\_insert](#), [\\_M\\_log\\_k](#), and [\\_M\\_losers](#).

#### `~_LoserTreeBase()`

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase () [inline]
```

The destructor.

References [\\_M\\_losers](#).

### 6.113.3 Member Function Documentation

#### `__get_min_source()`

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline]
```

#### Returns

the index of the sequence with the smallest element.

References [\\_M\\_losers](#).

#### `__insert_start()`

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                      |
| <code>__source</code> | <code>__index</code> of the <code>__source</code> sequence |

|                    |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------|
| <code>__sup</code> | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |
|--------------------|-------------------------------------------------------------------------------------------|

References [\\_M\\_first\\_insert](#), and [\\_M\\_losers](#).

#### 6.113.4 Member Data Documentation

##### `_M_comp`

```
template<typename _Tp, typename _Compare>
_Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected]
```

`_Compare` to use.  
Referenced by [\\_LoserTreeBase\(\)](#).

##### `_M_first_insert`

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected]
```

State flag that determines whether the `_LoserTree` is empty.  
Only used for building the `_LoserTree`.  
Referenced by [\\_LoserTreeBase\(\)](#), and [\\_\\_insert\\_start\(\)](#).

##### `_M_log_k`

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
```

`log_2[_M_k]`  
Referenced by [\\_LoserTreeBase\(\)](#).

##### `_M_losers`

```
template<typename _Tp, typename _Compare>
_Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers [protected]
```

`_LoserTree` elements.  
Referenced by [\\_LoserTreeBase\(\)](#), [~\\_LoserTreeBase\(\)](#), [\\_\\_get\\_min\\_source\(\)](#), and [\\_\\_insert\\_start\(\)](#).  
The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 6.114 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



#### Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

#### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

#### 6.114.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 6.115 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



### Public Member Functions

- `_LoserTreePointer` (unsigned int \_\_k, \_Compare \_\_comp=[std::less<\\_Tp>\(\)](#))
- void `__delete_min_insert` (const \_Tp &\_\_key, bool \_\_sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

### Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

#### 6.115.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation.

The stable variant is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 6.116 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`:



## Classes

- struct [\\_Loser](#)

## Public Member Functions

- **\_LoserTreePointerBase** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- [\\_Loser](#) \* **\_M\_losers**
- unsigned int **\_M\_offset**

### 6.116.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>
```

Base class of `_Loser` Tree implementation using pointers.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 6.117 `__gnu_parallel::_LoserTreePointerUnguarded<__stable, _Tp, _Compare>` Class Template Reference

```
#include <losertree.h>
```



Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



### Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 6.117.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
```

Stable unguarded \_LoserTree variant storing pointers.

Unstable variant is implemented below using partial specialization.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 6.118 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



#### Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (const \_Tp &\_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 6.118.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded \_LoserTree variant storing pointers.

Stable variant is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 6.119 \_\_gnu\_parallel::\_LoserTreePointerUnguardedBase< \_Tp, \_Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`:



### Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less<\\_Tp>](#) ())
- `int __get_min_source` ()
- `void __insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

### Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

#### 6.119.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 6.120 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

```
#include <multiway_merge.h>
```

### Static Public Attributes

- `static const bool _M_use_pointer`

### 6.120.1 Detailed Description

```
template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>_Tp</code> | type to give the loser tree traits for. |
|------------------|-----------------------------------------|

### 6.120.2 Member Data Documentation

#### `_M_use_pointer`

```
template<typename _Tp>
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer [static]
```

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

### 6.121 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



**Public Member Functions**

- **\_LoserTreeUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

**Protected Attributes**

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

**6.121.1 Detailed Description**

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded \_LoserTree.

Unstable variant is selected below with partial specialization.

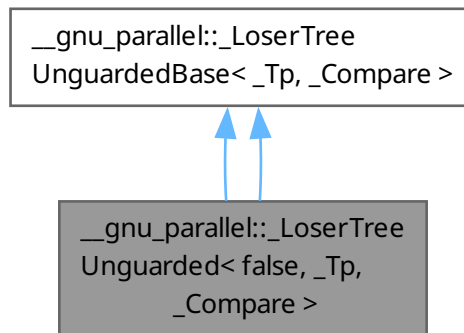
The documentation for this class was generated from the following file:

- [losertree.h](#)

**6.122 \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare > Class Template Reference**

```
#include <losertree.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare >:

**Public Member Functions**

- **\_LoserTreeUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()

- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

#### 6.122.1 Detailed Description

template<typename \_Tp, typename \_Compare>

class **\_\_gnu\_parallel::\_LoserTreeUnguarded**< false, \_Tp, \_Compare >

Non-Stable implementation of ungarded \_LoserTree.

Stable implementation is above.

The documentation for this class was generated from the following file:

- [losertree.h](#)

### 6.123 \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreeUnguardedBase< \_Tp, \_Compare >:



#### Public Member Functions

- **\_LoserTreeUnguardedBase** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

#### Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- \_Loser \* **\_M\_losers**
- unsigned int **\_M\_offset**

### 6.123.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >
```

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 6.124 `__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits< _Alloc, Metadata >::const_reference` **const\_reference**
- typedef `Metadata` **metadata\_type**

### Public Member Functions

- `const_reference` **get\_metadata** () const

### Public Attributes

- `metadata_type` **m\_metadata**

### 6.124.1 Detailed Description

```
template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
```

Metadata base primary template.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.125 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

```
#include <pat_trie_base.hpp>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits< _Alloc, null_type >::const_reference` **const\_reference**
- typedef `null_type` **metadata\_type**

### Public Member Functions

- `const_reference` **get\_metadata** () const

## Public Attributes

- metadata\_type **m\_metadata**

### 6.125.1 Detailed Description

```
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::Metadata< null_type, _Alloc >
```

Specialization for null metadata.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.126 \_\_gnu\_parallel::\_Multiplies< \_Tp1, \_Tp2, \_Result > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_Multiplies< \_Tp1, \_Tp2, \_Result >:



## Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

## Public Member Functions

- \_Result **operator()** (const \_Tp1 &\_\_x, const \_Tp2 &\_\_y) const

### 6.126.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0)))>
struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >
```

Similar to `std::multiplies`, but allows two different types.



### 6.126.2 Member Typedef Documentation

#### first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:
```

- [base.h](#)

## 6.127 std::\_\_fwlist::\_Node<\_ValPtr> Struct Template Reference

```
#include <forward_list.h>
```

Inheritance diagram for std::\_\_fwlist::\_Node<\_ValPtr>:



### Public Types

- using `_Base_ptr`
- using `_Node_ptr`
- using `value_type`

### Public Member Functions

- `_Node(_Node &&)=delete`

- `_Base_ptr _M_base_ptr () const`
- `_Node_ptr _M_node_ptr ()`
- `void _M_reverse_after () noexcept`
- `_Base_ptr _M_transfer_after (_Base_ptr __begin, _Base_ptr __end) noexcept`
- `const value_type * _M_valptr () const noexcept`
- `value_type * _M_valptr () noexcept`

#### Public Attributes

- `_Base_ptr _M_next`
- `_Uninit_storage _M_u`

#### 6.127.1 Detailed Description

```
template<typename _ValPtr>
struct std::__fwlist::_Node< _ValPtr >
```

A helper node class for `forward_list`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 6.128 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_ATraits, Metadata > Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >`:



#### Public Types

- `typedef _ATraits::const_iterator a_const_iterator`
- `typedef detail::rebind\_traits< _Alloc, _ATraits >::const_pointer a_const_pointer`

- typedef `_ATraits` `access_traits`
- typedef `_Alloc` `allocator_type`
- typedef `detail::rebind_traits<_Alloc, _Node_base>::pointer` `node_pointer`
- typedef `_ATraits::type_traits` `type_traits`

### Public Member Functions

- `_Node_base` (`node_type` type)

### Public Attributes

- node\_pointer `m_p_parent`
- const `node_type` `m_type`

#### 6.128.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::__Node_base<_ATraits, Metadata>
```

Node base.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.129 `std::__fwlist::__Node_base<_VoidPtr>` Struct Template Reference

```
#include <forward_list.h>
```

### Public Types

- using `_Base_ptr`

### Public Member Functions

- `_Node_base` (`_Node_base` &&\_\_x) noexcept
- `_Node_base` (const `_Node_base` &)=delete
- `_Base_ptr` `_M_base_ptr` () const
- void `_M_reverse_after` () noexcept
- `_Base_ptr` `_M_transfer_after` (`_Base_ptr` \_\_begin, `_Base_ptr` \_\_end) noexcept
- `_Node_base` & `operator=` (`_Node_base` &&\_\_x) noexcept
- `_Node_base` & `operator=` (const `_Node_base` &)=delete

### Public Attributes

- `_Base_ptr` `_M_next`

#### 6.129.1 Detailed Description

```
template<typename _VoidPtr>
struct std::__fwlist::__Node_base<_VoidPtr>
```

The node-base type for allocators that use fancy pointers.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

### 6.130 `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

#include <pat\_trie\_base.hpp>

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



#### Public Types

- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, [metadata\\_type](#) >::const\_reference [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef value\_type **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef \_CIterator **value\_type**

#### Public Member Functions

- **\_Node\_citer** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_citer** [get\\_child](#) (size\_type i) const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- size\_type [num\\_children](#) () const
- bool [operator!=](#) (const \_Node\_citer &other) const
- const\_reference [operator\\*](#) () const
- bool [operator==](#) (const \_Node\_citer &other) const
- [std::pair](#)< a\_const\_iterator, a\_const\_iterator > [valid\\_prefix](#) () const

#### Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

#### Protected Types

- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef Node::a\_const\_pointer **a\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::const\_pointer [inode\\_const\\_pointer](#)
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::pointer [inode\\_pointer](#)

- typedef `rebind_traits< _Alloc, Leaf >::const_pointer` `leaf_const_pointer`
- typedef `rebind_traits< _Alloc, Leaf >::pointer` `leaf_pointer`
- typedef `rebind_traits< _Alloc, Node >::pointer` `node_pointer`

### 6.130.1 Detailed Description

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`

`class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`

Node const iterator.

### 6.130.2 Member Typedef Documentation

#### `metadata_const_reference`

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<`  
`Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference`  
Const metadata reference type.

#### `metadata_type`

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,`  
`Inode, _CIterator, Iterator, _Alloc >::metadata_type`  
Metadata type.

### 6.130.3 Member Function Documentation

#### `get_child()`

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`_Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,`  
`Iterator, _Alloc >::get_child (`  
`size_type i) const [inline]`

Returns a `__const` node `__iterator` to the corresponding node's `i`-th child.

#### `get_metadata()`

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,`  
`_CIterator, Iterator, _Alloc >::get_metadata () const [inline]`  
Metadata access.

#### `num_children()`

`template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>`  
`size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,`  
`Iterator, _Alloc >::num_children () const [inline]`  
Returns the number of children in the corresponding node.

Referenced by `__gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*()`.

**operator!=(())**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!=(
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

**operator\*()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _C
Iterator, Iterator, _Alloc >::operator* () const [inline]
```

Const access; returns the `__const iterator*` associated with the current leaf.

**operator==(())**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator==(
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

**valid\_prefix()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.131 `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator,`

`_Alloc >`:



## Public Types

- typedef value\_type **const\_reference**
- typedef trivial\_iterator\_difference\_type **difference\_type**
- typedef trivial\_iterator\_tag **iterator\_category**
- typedef rebind\_traits< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Node::metadata\_type **metadata\_type**
- typedef value\_type **reference**
- typedef base\_type::size\_type **size\_type**
- typedef Iterator **value\_type**

## Public Member Functions

- **\_Node\_iter** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_iter** [get\\_child](#) (size\_type i) const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- size\_type [num\\_children](#) () const
- bool [operator!=](#) (const \_Node\_citer &other) const
- reference [operator\\*](#) () const
- bool [operator==](#) (const \_Node\_citer &other) const
- [std::pair](#)< a\_const\_iterator, a\_const\_iterator > [valid\\_prefix](#) () const

## Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

## Protected Types

- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef rebind\_traits< \_Alloc, Inode >::const\_pointer **inode\_const\_pointer**
- typedef rebind\_traits< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef rebind\_traits< \_Alloc, Leaf >::pointer **leaf\_pointer**

### 6.131.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
```

```
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node iterator.

### 6.131.2 Member Typedef Documentation

#### metadata\_const\_reference

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference [inherited]
Const metadata reference type.
```

#### metadata\_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]
Metadata type.
```

### 6.131.3 Member Function Documentation

#### get\_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
 size_type i) const [inline]
```

Returns a node \_\_iterator to the corresponding node's i-th child.

#### get\_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const [inline], [inherited]
Metadata access.
```

#### num\_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const [inline], [inherited]
```

Returns the number of children in the corresponding node.

Referenced by [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer](#)< node, leaf, head, inode, const\_iterator, iterator, \_Alloc >::operator\*().



**operator!=()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

**operator\*()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* () const [inline]
```

Access; returns the iterator\* associated with the current leaf.

**operator==(())**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator==(
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content to a different iterator object.

**valid\_prefix()**

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline], [inherited]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

**6.132 `__gnu_debug::_Not_equal_to<_Type>` Class Template Reference**

```
#include <safe_sequence.h>
```

**Public Member Functions**

- `constexpr _Not_equal_to (const _Type &__v)`
- `bool operator() (const _Type &__x) const`

**6.132.1 Detailed Description**

```
template<typename _Type>
class __gnu_debug::_Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

### 6.133 std::\_Not\_fn<\_Fn> Class Template Reference

```
#include <functional>
```

#### Public Member Functions

- `template<typename _Fn2>`  
`constexpr _Not_fn (_Fn2 && __fn, int)`
- `_Not_fn (_Not_fn && __fn)=default`
- `_Not_fn (const _Not_fn & __fn)=default`
- `template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &&, _Args...>::value>>`  
`void operator() (_Args &&... __args) &&=delete`
- `template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &&, _Args...>::value>>`  
`constexpr decltype(_S_not< __inv_res_t< _Fn &&, _Args... > >()) operator() (_Args &&... __args)`  
`&&noexcept(__is_nothrow_invocable< _Fn &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`&&, _Args... > >()))`
- `template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &, _Args...>::value>>`  
`void operator() (_Args &&... __args) &=delete`
- `template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &, _Args...>::value>>`  
`constexpr decltype(_S_not< __inv_res_t< _Fn &, _Args... > >()) operator() (_Args &&... __args) &noexcept(↵`  
`__is_nothrow_invocable< _Fn &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &, _Args... > >()))`
- `template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &&, _Args...>::value>>`  
`void operator() (_Args &&... __args) const &&=delete`
- `template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &&, _Args...>::value>>`  
`constexpr decltype(_S_not< __inv_res_t< _Fn const &&, _Args... > >()) operator() (_Args &&... __args) const`  
`&&noexcept(__is_nothrow_invocable< _Fn const &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`const &&, _Args... > >()))`
- `template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &, _Args...>::value>>`  
`void operator() (_Args &&... __args) const &=delete`
- `template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &, _Args...>::value>>`  
`constexpr decltype(_S_not< __inv_res_t< _Fn const &, _Args... > >()) operator() (_Args &&... __args) const`  
`&noexcept(__is_nothrow_invocable< _Fn const &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`const &, _Args... > >()))`

#### 6.133.1 Detailed Description

```
template<typename _Fn>
```

```
class std::_Not_fn<_Fn>
```

Generalized negator.

The documentation for this class was generated from the following file:

- [functional](#)

### 6.134 \_\_gnu\_parallel::\_Nothing Struct Reference

```
#include <for_each_selectors.h>
```

#### Public Member Functions

- `template<typename _It>`  
`void operator() (_It __i)`

### 6.134.1 Detailed Description

Functor doing nothing.

For some \_\_reduction tasks (this is not a function object, but is passed as \_\_selector \_\_dummy parameter.

### 6.134.2 Member Function Documentation

#### operator>()

```
template<typename _It>
void __gnu_parallel::_Nothing::operator() (
 _It __i) [inline]
```

Functor execution.

#### Parameters

|          |                              |
|----------|------------------------------|
| ↩        | iterator referencing object. |
| ↩        |                              |
| ↩        |                              |
| ↩        |                              |
| <i>i</i> |                              |

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 6.135 \_\_gnu\_parallel::\_Piece<\_DifferenceTp > Struct Template Reference

```
#include <multiway_mergesort.h>
```

### Public Types

- typedef \_DifferenceTp \_DifferenceType

### Public Attributes

- \_DifferenceType [\\_M\\_begin](#)
- \_DifferenceType [\\_M\\_end](#)

### 6.135.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Piece<_DifferenceTp >
```

Subsequence description.

### 6.135.2 Member Data Documentation

#### [\\_M\\_begin](#)

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece<_DifferenceTp >::_M_begin
```

Begin of subsequence.

**\_M\_end**

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
End of subsequence.
```

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

**6.136 std::\_Placeholder< \_Num > Struct Template Reference**

```
#include <functional>
```

**6.136.1 Detailed Description**

```
template<int _Num>
struct std::_Placeholder< _Num >
```

The type of placeholder objects defined by libstdc++.

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

**6.137 \_\_gnu\_parallel::\_Plus< \_Tp1, \_Tp2, \_Result > Struct Template Reference**

```
#include <base.h>
```

Inheritance diagram for \_\_gnu\_parallel::\_Plus< \_Tp1, \_Tp2, \_Result >:

**Public Types**

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

### 6.137.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0)))>
struct __gnu_parallel::Plus<_Tp1, _Tp2, _Result>
```

Similar to `std::plus`, but allows two different types.

### 6.137.2 Member Typedef Documentation

#### `first_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Result>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### `result_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function<_Arg1, _Arg2, _Result>::result_type [inherited]
result_type is the return type
```

#### `second_argument_type`

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Result>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [base.h](#)

## 6.138 `__gnu_parallel::__PMWSSortingData<_RAIter>` Struct Template Reference

```
#include <multiway_mergesort.h>
```

## Public Types

- typedef `_TraitsType::difference_type` `_DifferenceType`
- typedef `std::iterator_traits<_RAIter>` `_TraitsType`
- typedef `_TraitsType::value_type` `_ValueType`

## Public Attributes

- `_ThreadIndex` `_M_num_threads`
- `_DifferenceType` \* `_M_offsets`
- `std::vector<_Piece<_DifferenceType>>` \* `_M_pieces`
- `_ValueType` \* `_M_samples`
- `_RAIter` `_M_source`
- `_DifferenceType` \* `_M_starts`
- `_ValueType` \*\* `_M_temporary`

### 6.138.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_PMWMSSortingData< _RAIter >
```

Data accessed by all threads.

PMWMS = parallel multiway mergesort

### 6.138.2 Member Data Documentation

#### **\_M\_num\_threads**

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads
```

Number of threads involved.  
Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

#### **\_M\_offsets**

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_offsets
```

Offsets to add to the found positions.  
Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#).

#### **\_M\_pieces**

```
template<typename _RAIter>
std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces
```

Pieces of data to merge [thread][\_\_sequence].  
Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

#### **\_M\_samples**

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples
```

Samples.  
Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#).

#### **\_M\_source**

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source
```

Input \_\_begin.  
Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

#### **\_M\_starts**

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts
```

Start indices, per thread.  
Referenced by [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#).

#### **\_M\_temporary**

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary
```

Storage in which to sort.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 6.139 `__gnu_cxx::Pointer_adapter<_Storage_policy>` Class Template Reference

```
#include <pointer.h>
```

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Storage_policy::element_type` **element\_type**
- typedef `std::random_access_iterator_tag` **iterator\_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value\_type**

### Public Member Functions

- template<typename `_Up`>  
  **\_Pointer\_adapter** (`_Up` \*\_\_arg)
- **\_Pointer\_adapter** (const `_Pointer_adapter` &\_\_arg)
- template<typename `_Up`>  
  **\_Pointer\_adapter** (const `_Pointer_adapter`< `_Up` > &\_\_arg)
- **\_Pointer\_adapter** (`element_type` \*\_\_arg=0)
- **operator bool** () const
- `reference` **operator\*** () const
- `_Pointer_adapter` & **operator++** ()
- `_Pointer_adapter` **operator++** (int)
- `_Pointer_adapter` & **operator+=** (int \_\_offset)
- `_Pointer_adapter` & **operator+=** (long \_\_offset)
- `_Pointer_adapter` & **operator+=** (long long \_\_offset)
- `_Pointer_adapter` & **operator+=** (short \_\_offset)
- `_Pointer_adapter` & **operator+=** (unsigned int \_\_offset)
- `_Pointer_adapter` & **operator+=** (unsigned long \_\_offset)
- `_Pointer_adapter` & **operator+=** (unsigned long long \_\_offset)
- `_Pointer_adapter` & **operator+=** (unsigned short \_\_offset)
- template<typename `_Up`>  
  `std::ptrdiff_t` **operator-** (const `_Pointer_adapter`< `_Up` > &\_\_rhs) const
- `_Pointer_adapter` & **operator--** ()
- `_Pointer_adapter` **operator--** (int)
- `_Pointer_adapter` & **operator-=** (int \_\_offset)
- `_Pointer_adapter` & **operator-=** (long \_\_offset)
- `_Pointer_adapter` & **operator-=** (long long \_\_offset)
- `_Pointer_adapter` & **operator-=** (short \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned int \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned long \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned long long \_\_offset)
- `_Pointer_adapter` & **operator-=** (unsigned short \_\_offset)
- `element_type` \* **operator->** () const

- `template<typename _Up>`  
`_Pointer_adapter & operator= (_Up * __arg)`
- `_Pointer_adapter & operator= (const _Pointer_adapter & __arg)`
- `template<typename _Up>`  
`_Pointer_adapter & operator= (const _Pointer_adapter< _Up > & __arg)`
- `reference operator[] (std::ptrdiff_t __index) const`

## Friends

- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, int __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, long __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, long long __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, short __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, unsigned int __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, unsigned long __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, unsigned long long __offset)`
- `_Pointer_adapter operator+ (const _Pointer_adapter & __lhs, unsigned short __offset)`
- `_Pointer_adapter operator+ (int __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (long __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (long long __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (short __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (unsigned int __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (unsigned long __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (unsigned long long __offset, const _Pointer_adapter & __rhs)`
- `_Pointer_adapter operator+ (unsigned short __offset, const _Pointer_adapter & __rhs)`
- `template<typename _Up>`  
`std::ptrdiff_t operator- (_Up * __lhs, const _Pointer_adapter & __rhs)`
- `template<typename _Up>`  
`std::ptrdiff_t operator- (const _Pointer_adapter & __lhs, _Up * __rhs)`
- `std::ptrdiff_t operator- (const _Pointer_adapter & __lhs, element_type * __rhs)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, int __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, long __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, long long __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, short __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, unsigned int __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, unsigned long __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, unsigned long long __offset)`
- `_Pointer_adapter operator- (const _Pointer_adapter & __lhs, unsigned short __offset)`
- `std::ptrdiff_t operator- (element_type * __lhs, const _Pointer_adapter & __rhs)`
- `std::strong_ordering operator<=> (const _Pointer_adapter & __lhs, const _Pointer_adapter & __rhs) noexcept`

### 6.139.1 Detailed Description

`template<typename _Storage_policy>`

`class __gnu_cxx::_Pointer_adapter< _Storage_policy >`

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.



A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to `const` and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The `const` qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp>>>; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp>>>; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp>>>; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp>>>;
```

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 6.140 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

```
#include <base.h>
```

### Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**
- typedef `_PseudoSequenceIterator<_Tp, uint64_t>` **`iterator`**

### Public Member Functions

- `_PseudoSequence` (`const _Tp &__val, _DifferenceType __count`)
- iterator `begin` () `const`
- iterator `end` () `const`

#### 6.140.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

#### Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M_value</code> type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

#### 6.140.2 Constructor & Destructor Documentation

##### `_PseudoSequence()`

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::_PseudoSequence (
 const _Tp &__val,
 _DifferenceType __count) [inline]
```

Constructor.

**Parameters**

|                      |                             |
|----------------------|-----------------------------|
| <code>__val</code>   | Element of the sequence.    |
| <code>__count</code> | Number of (virtual) copies. |

**6.140.3 Member Function Documentation****begin()**

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const [inline]
Begin iterator.
```

**end()**

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const [inline]
End iterator.
```

The documentation for this class was generated from the following file:

- [base.h](#)

**6.141 \_\_gnu\_parallel::\_PseudoSequenceliterator< \_Tp, \_DifferenceTp > Class Template Reference**

```
#include <base.h>
```

**Public Types**

- typedef `_DifferenceTp` **\_DifferenceType**

**Public Member Functions**

- **\_PseudoSequenceliterator** (const `_Tp` &`__val`, `_DifferenceType` `__pos`)
- bool **operator!=** (const `_PseudoSequenceliterator` &`__i2`)
- const `_Tp` & **operator\*** () const
- `_PseudoSequenceliterator` & **operator++** ()
- `_PseudoSequenceliterator` **operator++** (int)
- `_DifferenceType` **operator-** (const `_PseudoSequenceliterator` &`__i2`)
- bool **operator==** (const `_PseudoSequenceliterator` &`__i2`)
- const `_Tp` & **operator[]** (`_DifferenceType`) const

**6.141.1 Detailed Description**

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequenceliterator< _Tp, _DifferenceTp >
```

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

**Parameters**

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M_value</code> type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

The documentation for this class was generated from the following file:

- [base.h](#)

## 6.142 `__gnu_parallel::__QSBThreadLocal<_RAIter>` Struct Template Reference

```
#include <balanced_quicksort.h>
```

### Public Types

- typedef `_TraitsType::difference_type` `_DifferenceType`
- typedef `std::pair<_RAIter, _RAIter>` `_Piece`
- typedef `std::iterator_traits<_RAIter>` `_TraitsType`

### Public Member Functions

- `_QSBThreadLocal` (int `__queue_size`)

### Public Attributes

- volatile `_DifferenceType` \* `_M_elements_leftover`
- `_Piece` `_M_global`
- `_Piece` `_M_initial`
- `_RestrictedBoundedConcurrentQueue<_Piece>` `_M_leftover_parts`
- `_ThreadIndex` `_M_num_threads`

#### 6.142.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::__QSBThreadLocal<_RAIter>
```

Information local to one thread in the parallel quicksort run.

#### 6.142.2 Member Typedef Documentation

##### `_Piece`

```
template<typename _RAIter>
typedef std::pair<_RAIter, _RAIter> __gnu_parallel::__QSBThreadLocal<_RAIter>::_Piece
```

Continuous part of the sequence, described by an iterator pair.

#### 6.142.3 Constructor & Destructor Documentation

##### `_QSBThreadLocal()`

```
template<typename _RAIter>
__gnu_parallel::__QSBThreadLocal<_RAIter>::_QSBThreadLocal (
 int __queue_size) [inline]
```

Constructor.

##### Parameters

|                           |                                  |
|---------------------------|----------------------------------|
| <code>__queue_size</code> | size of the work-stealing queue. |
|---------------------------|----------------------------------|

References `_M_leftover_parts`.

#### 6.142.4 Member Data Documentation

##### **`_M_elements_leftover`**

```
template<typename _RAIter>
```

```
volatile _DifferenceType* __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

##### **`_M_global`**

```
template<typename _RAIter>
```

```
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

##### **`_M_initial`**

```
template<typename _RAIter>
```

```
_Piece __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

##### **`_M_leftover_parts`**

```
template<typename _RAIter>
```

```
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_↵
```

leftover\_parts

Work-stealing queue.

Referenced by [\\_QSBThreadLocal\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

##### **`_M_num_threads`**

```
template<typename _RAIter>
```

```
_ThreadIndex __gnu_parallel::__QSBThreadLocal< _RAIter >::_M_num_threads
```

Number of threads involved in this algorithm.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#).

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

### 6.143 `std::__detail::_Quoted_string< _String, _CharT >` Struct Template Reference

```
#include <quoted_string.h>
```

#### Public Member Functions

- `_Quoted_string` (`_String __str, _CharT __del, _CharT __esc`)
- `_Quoted_string & operator=` (`_Quoted_string &`)=delete

#### Public Attributes

- `_CharT _M_delim`
- `_CharT _M_escape`
- `_String _M_string`

## Friends

- `template<typename _CharT2, typename _Traits, typename _String2>  
std::basic_ostream< _CharT2, _Traits > & operator<< (std::basic_ostream< _CharT2, _Traits > &__os, const  
_Quoted_string< _String2, _CharT2 > &__str)`
- `template<typename _CharT2, typename _Traits>  
std::basic_ostream< _CharT2, _Traits > & operator<< (std::basic_ostream< _CharT2, _Traits > &__os, const  
_Quoted_string< const _CharT2 *, _CharT2 > &__str)`
- `template<typename _CharT2, typename _Traits, typename _Alloc>  
std::basic_istream< _CharT2, _Traits > & operator>> (std::basic_istream< _CharT2, _Traits > &__is, const  
_Quoted_string< basic_string< _CharT2, _Traits, _Alloc > &, _CharT2 > &__str)`

### 6.143.1 Detailed Description

`template<typename _String, typename _CharT>`  
`struct std::__detail::_Quoted_string< _String, _CharT >`

Struct for delimited strings.

The documentation for this struct was generated from the following file:

- [quoted\\_string.h](#)

## 6.144 \_\_gnu\_parallel::\_RandomNumber Class Reference

`#include <random_number.h>`

### Public Member Functions

- [\\_RandomNumber](#) ()
- [\\_RandomNumber](#) (uint32\_t \_\_seed, uint64\_t \_M\_supremum=0x100000000ULL)
- unsigned long [\\_\\_genrand\\_bits](#) (int \_\_bits)
- uint32\_t [operator\(\)](#) ()
- uint32\_t [operator\(\)](#) (uint64\_t local\_supremum)

### 6.144.1 Detailed Description

Random number generator, based on the Mersenne twister.

### 6.144.2 Constructor & Destructor Documentation

**[\\_RandomNumber\(\)](#)** [1/2]

`__gnu_parallel::_RandomNumber::_RandomNumber () [inline]`

Default constructor. Seed with 0.

**[\\_RandomNumber\(\)](#)** [2/2]

`__gnu_parallel::_RandomNumber::_RandomNumber (  
    uint32_t __seed,  
    uint64_t _M_supremum = 0x100000000ULL) [inline]`

Constructor.

### Parameters

|                     |                |
|---------------------|----------------|
| <code>__seed</code> | Random __seed. |
|---------------------|----------------|

|                           |                                                                    |
|---------------------------|--------------------------------------------------------------------|
| <code>__M_supremum</code> | Generate integer random numbers in the interval [0, __M_supremum). |
|---------------------------|--------------------------------------------------------------------|

### 6.144.3 Member Function Documentation

#### `__genrand_bits()`

```
unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (
 int __bits) [inline]
```

Generate a number of random bits, run-time parameter.

#### Parameters

|                     |                             |
|---------------------|-----------------------------|
| <code>__bits</code> | Number of bits to generate. |
|---------------------|-----------------------------|

#### `operator>()` [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]
```

Generate unsigned random 32-bit integer.

#### `operator>()` [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
 uint64_t local_supremum) [inline]
```

Generate unsigned random 32-bit integer in the interval [0,local\_supremum).

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

### 6.145 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

```
#include <pointer.h>
```

Inheritance diagram for `__gnu_cxx::_Relative_pointer_impl<_Tp>`:



### Public Types

- typedef `_Tp` **element\_type**

### Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Relative\_pointer\_impl &__arg) const`
- `bool operator== (const \_Relative\_pointer\_impl &__arg) const`
- `void set (_Tp *__arg)`

#### 6.145.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 6.146 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

```
#include <pointer.h>
```

Inheritance diagram for `__gnu_cxx::_Relative_pointer_impl< const _Tp >`:



### Public Types

- typedef const \_Tp **element\_type**

### Public Member Functions

- const \_Tp \* **get** () const
- bool **operator**< (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- bool **operator**== (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- void **set** (const \_Tp \*\_\_arg)

#### 6.146.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const _Tp >
```

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic. The documentation for this class was generated from the following file:

- [pointer.h](#)

## 6.147 `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >` Class Template Reference

```
#include <queue.h>
```

### Public Member Functions

- [\\_RestrictedBoundedConcurrentQueue](#) ([\\_SequenceIndex](#) \_\_max\_size)
- [~\\_RestrictedBoundedConcurrentQueue](#) ()



- bool `pop_back` (`_Tp &__t`)
- bool `pop_front` (`_Tp &__t`)
- void `push_front` (`const _Tp &__t`)

### 6.147.1 Detailed Description

`template<typename _Tp>`  
**class** `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>_Tp</code> | Contained element type. |
|------------------|-------------------------|

### 6.147.2 Constructor & Destructor Documentation

#### `_RestrictedBoundedConcurrentQueue()`

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::_RestrictedBoundedConcurrentQueue (
 __SequenceIndex __max_size) [inline]
```

Constructor. Not to be called concurrent, of course.

#### Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__max_size</code> | Maximal number of elements to be contained. |
|-------------------------|---------------------------------------------|

References `__gnu_parallel::__encode2()`.

#### `~_RestrictedBoundedConcurrentQueue()`

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::~~_RestrictedBoundedConcurrentQueue ()
[inline]
```

Destructor. Not to be called concurrent, of course.

### 6.147.3 Member Function Documentation

#### `pop_back()`

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_back (
 _Tp &__t) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

References `__gnu_parallel::__compare_and_swap()`, `__gnu_parallel::__decode2()`, and `__gnu_parallel::__encode2()`.

#### `pop_front()`

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_front (
 _Tp &__t) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with `pop_front()`.

References [\\_\\_gnu\\_parallel::\\_\\_compare\\_and\\_swap\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_decode2\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#).

### **push\_front()**

```
template<typename _Tp>
void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
 const _Tp & __t) [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with `pop_front()`.

References [\\_\\_gnu\\_parallel::\\_\\_decode2\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_encode2\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_fetch\\_and\\_add\(\)](#).

The documentation for this class was generated from the following file:

- [queue.h](#)

## **6.148 \_\_gnu\_debug::\_Safe\_container< \_SafeContainer, \_Alloc, \_SafeBase, \_IsCxx11AllocatorAware > Class Template Reference**

```
#include <safe_container.h>
```

Inheritance diagram for `__gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware`

>:



## Protected Member Functions

- `_Safe_container` (`_Safe_container &&`)=default
- `constexpr _Safe_container` (`_Safe_container &&__x, const _Alloc &__a`)
- `_Safe_container` (`const _Safe_container &`)=default

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept
- constexpr \_Safe\_container & **operator=** (\_Safe\_container &&\_\_x) noexcept
- constexpr \_Safe\_container & **operator=** (const \_Safe\_container &) noexcept=default

#### 6.148.1 Detailed Description

```
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >
```

Safe class dealing with some allocator dependent operations.

The documentation for this class was generated from the following file:

- [safe\\_container.h](#)

### 6.149 \_\_gnu\_debug::\_Safe\_forward\_list< \_SafeSequence > Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for \_\_gnu\_debug::\_Safe\_forward\_list< \_SafeSequence >:



#### Public Member Functions

- constexpr void **\_M\_invalidate\_if** (\_Predicate \_\_pred) const
- void **\_M\_transfer\_from\_if** (const \_Safe\_sequence &\_\_from, \_Predicate \_\_pred) const

#### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void **\_M\_detach\_all** () const
- void **\_M\_detach\_singular** () const
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () const noexcept
- void **\_M\_invalidate\_all** () const
- void **\_M\_revalidate\_singular** () const
- void **\_M\_swap** (const \_Safe\_forward\_list &) const noexcept
- void **\_M\_swap** (const \_Safe\_sequence\_base &\_\_x) const noexcept

#### 6.149.1 Detailed Description

```
template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list< _SafeSequence >
```

Special iterators swap and invalidation for forward\_list because of the before\_begin iterator.

### 6.149.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

#### `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence<_SafeSequence>::_M_invalidate_if (
 _Predicate __pred) const [inline], [constexpr], [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence<_SafeSequence>::_M_transfer_from_if (
 const _Safe_sequence<_SafeSequence> & __from,
 _Predicate __pred) const [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References [\\_M\\_iterators](#).

### 6.149.3 Member Data Documentation

#### `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

## **\_M\_iterators**

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by [\\_M\\_transfer\\_from\\_if\(\)](#).

## **\_M\_version**

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

## **6.150 \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence, \_Category> Class Template Reference**

```
#include <safe_iterator.h>
```

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>`:



## **Public Types**

- `typedef _Safe_iterator<_Iterator, _Sequence, iterator_category> _Self`
- `typedef _Traits::difference_type difference_type`
- `typedef _Traits::iterator_category iterator_category`
- `using iterator_concept`
- `typedef _Iterator iterator_type`
- `typedef _Traits::pointer pointer`
- `typedef _Traits::reference reference`
- `typedef _Traits::value_type value_type`

## **Public Member Functions**

- `constexpr \_Safe\_iterator () noexcept`
- `constexpr \_Safe\_iterator (_Iterator __i, const _Safe_sequence_base *__seq) noexcept`
- `constexpr \_Safe\_iterator (_Safe_iterator &&__x) noexcept`

- constexpr `_Safe_iterator` (const `_Safe_iterator` &\_\_x) noexcept
- template<typename `_MutableIterator`>  
constexpr `_Safe_iterator` (const `_Safe_iterator`< `_MutableIterator`, `_Sequence`, typename `__gnu_cxx::__enable_if<_IsConstant::__value &&std::__are_same<_MutableIterator, _OtherIterator>::__value, _Category>::__type` > &\_\_x) noexcept
- void `_M_attach` (const `_Safe_sequence_base` \* \_\_seq)
- void `_M_attach_single` (const `_Safe_sequence_base` \* \_\_seq)
- bool `_M_attached_to` (const `_Safe_sequence_base` \* \_\_seq) const
- bool `_M_before_dereferenceable` () const
- template<typename `_Diff`>  
bool `_M_can_advance` (const `std::pair`< `_Diff`, `_Distance_precision` > &\_\_dist, int \_\_way) const
- bool `_M_can_advance` (difference\_type \_\_n, bool \_\_strict=false) const
- bool `_M_can_compare` (const `_Safe_iterator_base` &\_\_x) const noexcept
- bool `_M_dereferenceable` () const
- void `_M_detach_single` () noexcept
- `_Distance_traits`< `_Iterator` >::\_\_type `_M_get_distance_from_begin` () const
- `_Distance_traits`< `_Iterator` >::\_\_type `_M_get_distance_to` (const `_Safe_iterator` &\_\_rhs) const
- `_Distance_traits`< `_Iterator` >::\_\_type `_M_get_distance_to_end` () const
- `__gnu_cxx::__conditional_type`< `_IsConstant::__value`, const `_Sequence` \*, `_Sequence` \* >::\_\_type `_M_get_sequence` () const
- bool `_M_incrementable` () const
- void `_M_invalidate` ()
- bool `_M_is_before_begin` () const
- constexpr bool `_M_is_begin` () const
- bool `_M_is_beginnest` () const
- bool `_M_is_end` () const
- void `_M_reset` () noexcept
- bool `_M_singular` () const noexcept
- void `_M_unlink` () noexcept
- bool `_M_valid_range` (const `_Safe_iterator` &\_\_rhs, `std::pair`< difference\_type, `_Distance_precision` > &\_\_dist, bool \_\_check\_dereferenceable=true) const
- bool `_M_value_initialized` () const
- constexpr const `_Iterator` & `base` () const noexcept
- constexpr `_Iterator` & `base` () noexcept
- constexpr `operator _Iterator` () const noexcept
- constexpr reference `operator*` () const noexcept
- constexpr `_Safe_iterator` & `operator++` () noexcept
- constexpr `_Safe_iterator` `operator++` (int) noexcept
- constexpr pointer `operator->` () const noexcept
- constexpr `_Safe_iterator` & `operator=` (`_Safe_iterator` &&\_\_x) noexcept
- constexpr `_Safe_iterator` & `operator=` (const `_Safe_iterator` &\_\_x) noexcept

### Static Public Member Functions

- static constexpr bool `_S_constant` ()

### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- const `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

## Protected Types

- typedef std::\_\_are\_same< typename \_Sequence::\_\_Base::const\_iterator, \_Iterator > **\_IsConstant**
- typedef \_\_gnu\_cxx::\_\_conditional\_type< \_IsConstant::\_\_value, typename \_Sequence::\_\_Base::iterator, typename \_Sequence::\_\_Base::const\_iterator >::\_\_type **\_OtherIterator**

## Protected Member Functions

- constexpr **\_Safe\_iterator** (const \_Safe\_iterator &\_\_x, \_Unchecked) noexcept
- void **\_M\_attach** (const \_Safe\_sequence\_base \* \_\_seq, bool \_\_constant)
- void **\_M\_attach\_single** (const \_Safe\_sequence\_base \* \_\_seq, bool \_\_constant) noexcept
- void **\_M\_detach** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () noexcept

## Friends

- template<typename \_IterR>  
constexpr friend bool **operator==** (const \_Self & \_\_lhs, const \_Safe\_iterator< \_IterR, \_Sequence, iterator\_category > & \_\_rhs) noexcept
- constexpr friend bool **operator==** (const \_Self & \_\_lhs, const \_Self & \_\_rhs) noexcept

### 6.150.1 Detailed Description

template<typename \_Iterator, typename \_Sequence, typename \_Category = typename std::iterator\_traits<\_Iterator>::iterator\_category>

class \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, \_Category >

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Note that `_Iterator` must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before `_Iterator` get destroyed. Otherwise it would result in a data race.

### 6.150.2 Constructor & Destructor Documentation

**\_Safe\_iterator()** [1/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator () [inline],
[constexpr], [noexcept]
```

**Postcondition**

the iterator is singular and unattached

**\_Safe\_iterator()** [2/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
 _Iterator __i,
 const _Safe_sequence_base * __seq) [inline], [constexpr], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.



**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

**`_Safe_iterator()` [3/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
 const _Safe_iterator<_Iterator, _Sequence, _Category> & __x) [inline], [constexpr],
[noexcept]
Copy construction.
```

**`_Safe_iterator()` [4/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
 _Safe_iterator<_Iterator, _Sequence, _Category> && __x) [inline], [constexpr],
[noexcept]
Move construction.
```

**Postcondition**

`__x` is singular and unattached

**`_Safe_iterator()` [5/5]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
template<typename _MutableIterator>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_Safe_iterator (
 const _Safe_iterator<_MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<
_IsConstant::__value &&std::__are_same<_MutableIterator, _OtherIterator>::__value, _Category>::__type> & __x) [inline], [constexpr], [noexcept]
Converting constructor from a mutable iterator to a constant iterator.
```

**6.150.3 Member Function Documentation****`_M_attach()` [1/2]**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_M_attach (
 const _Safe_sequence_base * __seq) [inline]
Attach iterator to the given sequence.
```

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()` and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`.

**\_M\_attach()** [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**\_M\_attach\_single()** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >::_M_attach_single (
 const _Safe_sequence_base * __seq) [inline]
```

Likewise, but not thread-safe.

**\_M\_attach\_single()** [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**\_M\_attached\_to()**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).

**\_M\_before\_dereferenceable()**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >::_M_before_dereferenceable (
 const [inline]
```

Is the iterator before a dereferenceable one?

**\_M\_can\_compare()**

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const _Safe_iterator_base & __x) const [noexcept], [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References [\\_Safe\\_iterator\\_base\(\)](#), and [\\_M\\_can\\_compare\(\)](#).

Referenced by [\\_M\\_can\\_compare\(\)](#).

**`_M_dereferenceable()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator*()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, m`

**`_M_detach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.

References `_M_detach()`.

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator()`, `_M_detach()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`.

**`_M_detach_single()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () [noexcept], [inherited]
```

Likewise, but not thread-safe.

**`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept],
[inherited]
```

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`, `__gnu_debug::_Safe_local_iterator< _OtherIterator, map >::operator++()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=()`, `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`, and `__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=()`.

**`_M_incrementable()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_incrementable () const
[inline]
```

Is the iterator incrementable?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`, and `__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++()`.

**`_M_invalidate()`**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

References `_M_invalidate()`, and `_M_version`.

Referenced by `_M_invalidate()`.

**`_M_is_before_begin()`**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_before_begin () const
[inline]
```

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Referenced by `__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable()`.



Is the iterator value-initialized?

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`.

### `_S_constant()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
constexpr bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::_S_constant ()
[inline], [static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_attach()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_attach()`.

### `base()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Iterator & __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::base () [inline],
[constexpr], [noexcept]
```

Return the underlying iterator.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_is_begin()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_is_end()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_M_value_initialized()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator++()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator--()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator+=()`, and `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator-=()`.

### `operator_iterator()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator_iterator () const [inline],
[constexpr], [noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

### `operator*()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
reference __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator* () const
[inline], [nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

#### Precondition

iterator is dereferenceable

### `operator++()` [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator++ ()
[inline], [constexpr], [noexcept]
```

Iterator preincrement.

**Precondition**

iterator is incrementable

**operator++()** [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ (
 int) [inline], [constexpr], [noexcept]
```

Iterator postincrement.

**Precondition**

iterator is incrementable

**operator->()**

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator-> () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

**Precondition**

iterator is dereferenceable

**operator=()** [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
 _Safe_iterator< _Iterator, _Sequence, _Category > && __x) [inline], [constexpr],
[noexcept]
```

Move assignment.

**Postcondition**

\_\_x is singular and unattached

**operator=()** [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __x) [inline], [constexpr],
[noexcept]
```

Copy assignment.

**6.150.4 Member Data Documentation****M\_next**

[\\_Safe\\_iterator\\_base](#)\* [\\_\\_gnu\\_debug::\\_Safe\\_iterator\\_base::M\\_next](#) [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >::\\_M](#) and [\\_M\\_unlink\(\)](#).

**`_M_prior`**

`_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M` and `_M_unlink()`.

**`_M_sequence`**

`const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be `NULL` to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::_Safe_iterator()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_M_attached_to()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator`

**`_M_version`**

`unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_Safe_iterator_base()`, `_M_invalidate()`, and `__gnu_debug::_Safe_sequence`

The documentation for this class was generated from the following files:

- `boost_concept_check.h`
- `safe_iterator.h`
- `safe_iterator.tcc`

**6.151 `__gnu_debug::_Safe_iterator_base` Class Reference**

```
#include <safe_base.h>
```

Inheritance diagram for `__gnu_debug::__Safe_iterator_base`:



## Public Member Functions

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_compare (const _Safe_iterator_base & __x) const noexcept`



- void `_M_detach_single` () noexcept
- void `_M_invalidate` ()
- void `_M_reset` () noexcept
- bool `_M_singular` () const noexcept
- void `_M_unlink` () noexcept

### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- const `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

### Protected Member Functions

- constexpr `_Safe_iterator_base` ()
- constexpr `_Safe_iterator_base` (const `_Safe_iterator_base` & \_\_x, bool \_\_constant)
- constexpr `_Safe_iterator_base` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach_single` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant) noexcept
- void `_M_detach` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () noexcept

### Friends

- class `_Safe_sequence_base`

#### 6.151.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

#### 6.151.2 Constructor & Destructor Documentation

##### `_Safe_iterator_base`() [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base () [inline], [constexpr], [protected]
```

Initializes the iterator and makes it singular.

References `_M_next`, `_M_prior`, `_M_sequence`, and `_M_version`.

Referenced by `_Safe_iterator_base()`, and `_M_can_compare()`.

##### `_Safe_iterator_base`() [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
 const _Safe_sequence_base * __seq,
 bool __constant) [inline], [constexpr], [protected]
```

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

References `_M_attach()`, `_M_next`, `_M_prior`, `_M_sequence`, and `_M_version`.

### **`_Safe_iterator_base()`** [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
 const _Safe_iterator_base & __x,
 bool __constant) [inline], [constexpr], [protected]
```

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

References [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attach\(\)](#), [\\_M\\_next](#), [\\_M\\_prior](#), [\\_M\\_sequence](#), and [\\_M\\_version](#).

## **6.151.3 Member Function Documentation**

### **`_M_attach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

### **`_M_attach_single()`**

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [noexcept]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

### **`_M_attached_to()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).

### **`_M_can_compare()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const _Safe_iterator_base & __x) const [noexcept]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References [\\_Safe\\_iterator\\_base\(\)](#), and [\\_M\\_can\\_compare\(\)](#).

Referenced by [\\_M\\_can\\_compare\(\)](#).

### **`_M_detach()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.

References [\\_M\\_detach\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_M\\_detach\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#).

**`_M_detach_single()`**

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () [noexcept]
```

Likewise, but not thread-safe.

**`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept]
```

For use in `_Safe_iterator`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator++\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_invalidate()`**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline]
```

Invalidate the iterator, making it singular.

References [\\_M\\_invalidate\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\(\)](#).

**`_M_reset()`**

```
void __gnu_debug::_Safe_iterator_base::_M_reset () [noexcept]
```

Reset all member variables

**`_M_singular()`**

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const [noexcept]
```

Is this iterator singular?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_check\\_singular\\_aux\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::\\_M\\_incrementable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_unlink()`**

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept]
```

Unlink itself

References [\\_M\\_next](#), [\\_M\\_prior](#), and [\\_M\\_unlink\(\)](#).

Referenced by [\\_M\\_unlink\(\)](#).

**6.151.4 Member Data Documentation****`_M_next`**

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence >::\\_M\\_next\(\)](#), and [\\_M\\_unlink\(\)](#).

## \_M\_prior

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence` `!=` `NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M](#)  
and [\\_M\\_unlink\(\)](#).

## \_M\_sequence

```
const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by [\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, map >:: Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, m](#)  
[\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, map >:: Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#),  
[\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug:: Safe\\_sequence< \\_Sequence >:: M\\_transfer\\_from\\_if\(\)](#),  
[\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, map >:: operator=\(\)](#), [\\_\\_gnu\\_debug:: Safe\\_iterator< \\_Base\\_iterator, map >:: operator=\(\)](#),  
[\\_\\_gnu\\_debug:: Safe\\_local\\_iterator< \\_OtherIterator, UContainer >:: operator=\(\)](#), and [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator< \\_OtherIterato](#)

## \_M\_version

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `M` sequence for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence](#)

The documentation for this class was generated from the following file:

- `safe base.h`

## 6.152 \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_UContainer > Class Template Reference

```
#include <safe_local_iterator.h>
```

Inheritance diagram for `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>`:



### Public Types

- `typedef _Traits::difference_type` **difference\_type**
- `typedef _Traits::iterator_category` **iterator\_category**
- `typedef _Iterator` **iterator\_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value\_type**

### Public Member Functions

- `_Safe_local_iterator` () noexcept
- `_Safe_local_iterator` (\_Iterator \_\_i, const `_Safe_unordered_container_base` \*\_\_cont)
- `_Safe_local_iterator` (\_Safe\_local\_iterator &&\_\_x) noexcept
- `_Safe_local_iterator` (const `_Safe_local_iterator` &\_\_x) noexcept
- `template<typename _MutableIterator>`  
`_Safe_local_iterator` (const `_Safe_local_iterator`< `_MutableIterator`, typename `__gnu_cxx::__enable_if`< `_IsConstant`::\_\_value &&std::is\_same< `_MutableIterator`, `_OtherIterator` >::\_\_value, `_UContainer` >::\_\_type > &\_\_x) noexcept
- `void` `_M_attach` (const `_Safe_unordered_container_base` \*\_\_cont)
- `void` `_M_attach_single` (const `_Safe_unordered_container_base` \*\_\_cont)
- `bool` `_M_attached_to` (const `_Safe_sequence_base` \*\_\_seq) const
- `bool` `_M_can_compare` (const `_Safe_iterator_base` &\_\_x) const noexcept
- `bool` `_M_dereferenceable` () const
- `_Distance_traits`< `_Iterator` >::\_\_type `_M_get_distance_to` (const `_Safe_local_iterator` &\_\_rhs) const
- `std::conditional_t`< `_IsConstant`::\_\_value, const `_UContainer` \*, `_UContainer` \* > `_M_get_ucontainer` () const

- `template<typename _Other>`  
`bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _UContainer > &__other) const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `void _M_reset () noexcept`
- `const _Safe_unordered_container_base * _M_safe_container () const noexcept`
- `bool _M_singular () const noexcept`
- `void _M_unlink () noexcept`
- `bool _M_valid_range (const _Safe_local_iterator &__rhs, std::pair< difference_type, \_Distance\_precision > &__dist_info) const`
- `bool _M_value_initialized () const`
- `const _Iterator & base () const noexcept`
- `_Iterator & base () noexcept`
- `size_type bucket () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_local_iterator & operator++ ()`
- `_Safe_local_iterator operator++ (int)`
- `pointer operator-> () const`
- `_Safe_local_iterator & operator= (_Safe_local_iterator &&__x) noexcept`
- `_Safe_local_iterator & operator= (const _Safe_local_iterator &__x)`

### Static Public Member Functions

- `static constexpr bool _S_constant ()`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `const _Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_attach (const _Safe_sequence_base *__seq, bool __constant)`
- `void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)`
- `void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept`
- `void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept`
- `void _M_detach ()`
- `void _M_detach_single () noexcept`
- `__gnu_cxx::__mutex & _M_get_mutex () noexcept`

### Friends

- `bool operator!= (const _Self &__lhs, const _OtherSelf &__rhs) noexcept`
- `bool operator!= (const _Self &__lhs, const _Self &__rhs) noexcept`
- `bool operator== (const _Self &__lhs, const _OtherSelf &__rhs) noexcept`
- `bool operator== (const _Self &__lhs, const _Self &__rhs) noexcept`

### 6.152.1 Detailed Description

```
template<typename _Iterator, typename _UContainer>
class __gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >
```

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among unordered containers and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from unordered containers, and querying the iterator's state.

### 6.152.2 Constructor & Destructor Documentation

#### `_Safe_local_iterator()` [1/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator () [inline],
[noexcept]
```

##### Postcondition

the iterator is singular and unattached

#### `_Safe_local_iterator()` [2/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
 _Iterator __i,
 const __Safe_unordered_container_base * __cont) [inline]
```

Safe iterator construction from an unsafe iterator and its unordered container.

##### Precondition

`cont` is not NULL

##### Postcondition

this is not singular

#### `_Safe_local_iterator()` [3/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
 const __Safe_local_iterator< _Iterator, _UContainer > & __x) [inline], [noexcept]
```

Copy construction.

#### `_Safe_local_iterator()` [4/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
 __Safe_local_iterator< _Iterator, _UContainer > && __x) [inline], [noexcept]
```

Move construction.

##### Postcondition

`__x` is singular and unattached

**\_Safe\_local\_iterator()** [5/5]

```
template<typename _Iterator, typename _UContainer>
template<typename _MutableIterator>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
 const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _↵
IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer
>::__type > & __x) [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

**6.152.3 Member Function Documentation****\_M\_attach()** [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**\_M\_attach()** [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach (
 const _Safe_unordered_container_base * __cont) [inline]
```

Attach iterator to the given unordered container.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**\_M\_attach()** [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
 const _Safe_unordered_container_base * __cont,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by [\\_Safe\\_local\\_iterator\\_base\(\)](#), [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**\_M\_attach\_single()** [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**\_M\_attach\_single()** [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach_single (
 const _Safe_unordered_container_base * __cont) [inline]
```



Likewise, but not thread-safe.

### `_M_attach_single()` [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
 const __Safe_unordered_container_base * __cont,
 bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::_M_attach_single()`.

### `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const __Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References `_M_attached_to()`, and `_M_sequence`.

Referenced by `_M_attached_to()`.

### `_M_can_compare()`

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const __Safe_iterator_base & __x) const [noexcept], [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

References `_Safe_iterator_base()`, and `_M_can_compare()`.

Referenced by `_M_can_compare()`.

### `_M_dereferenceable()`

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator*()`, and `__gnu_debug::_Safe_local_iterator`.

### `_M_detach()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.

Referenced by `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::_Safe_local_iterator()`, `__gnu_debug::_Safe_local_iterator`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

### `_M_detach_single()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single () [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept],
[inherited]
```

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator++()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, __gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator`.

### **\_M\_in\_same\_bucket()**

```
template<typename _Iterator, typename _UContainer>
template<typename _Other>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_in_same_bucket (
 const _Safe_local_iterator< _Other, _UContainer > & __other) const [inline]
```

Is this iterator part of the same bucket as the other one?

### **\_M\_incrementable()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_incrementable () const [inline]
```

Is the iterator incrementable?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator++\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)

### **\_M\_invalidate()**

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

References [\\_M\\_invalidate\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\(\)](#).

### **\_M\_is\_begin()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_begin () const [inline]
```

Is this iterator equal to the container's begin(bucket) iterator?

### **\_M\_is\_end()**

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_end () const [inline]
```

Is this iterator equal to the container's end(bucket) iterator?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_dereferenceable\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_incrementable\(\)](#).

### **\_M\_reset()**

```
void __gnu_debug::_Safe_iterator_base::_M_reset () [noexcept], [inherited]
```

Reset all member variables

### **\_M\_singular()**

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const [noexcept], [inherited]
```

Is this iterator singular?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_check\\_singular\\_aux\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::\\_M\\_dereferenceable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::\\_M\\_incrementable\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator< \\_Base\\_iterator, map >::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator< \\_OtherIterator, \\_UContainer >::operator=\(\)](#).

**`_M_unlink()`**

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References [\\_M\\_next](#), [\\_M\\_prior](#), and [\\_M\\_unlink\(\)](#).

Referenced by [\\_M\\_unlink\(\)](#).

**`_M_value_initialized()`**

```
template<typename _Iterator, typename _UContainer>
```

```
bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_M_value_initialized () const [inline]
```

Is the iterator value-initialized?

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`_S_constant()`**

```
template<typename _Iterator, typename _UContainer>
```

```
constexpr bool __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::_S_constant () [inline], [static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_attach\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`base()`**

```
template<typename _Iterator, typename _UContainer>
```

```
_Iterator & __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::base () [inline], [noexcept]
```

Return the underlying iterator.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_Safe\\_local\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::bucket\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator++\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

**`bucket()`**

```
template<typename _Iterator, typename _UContainer>
```

```
size_type __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::bucket () const [inline]
```

Return the bucket.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_in\\_same\\_bucket\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::\\_M\\_is\\_end\(\)](#).

**`operator_iterator()`**

```
template<typename _Iterator, typename _UContainer>
```

```
__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::operator_iterator () const [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

**`operator*()`**

```
template<typename _Iterator, typename _UContainer>
```

```
reference __gnu_debug::_Safe_local_iterator<_Iterator, _UContainer>::operator* () const [inline]
```

Iterator dereference.

**Precondition**

iterator is dereferenceable

**operator++()** [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ ()
[inline]
```

Iterator preincrement.

**Precondition**

iterator is incrementable

**operator++()** [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ (
 int) [inline]
```

Iterator postincrement.

**Precondition**

iterator is incrementable

**operator->()**

```
template<typename _Iterator, typename _UContainer>
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator-> () const [inline]
```

Iterator dereference.

**Precondition**

iterator is dereferenceable

**operator=()** [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
 _Safe_local_iterator< _Iterator, _UContainer > && __x) [inline], [noexcept]
```

Move assignment.

**Postcondition**

\_\_x is singular and unattached

**operator=()** [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
 const _Safe_local_iterator< _Iterator, _UContainer > & __x) [inline]
```

Copy assignment.

#### 6.152.4 Member Data Documentation

##### `_M_next`

`_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_next` [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M](#) and [\\_M\\_unlink\(\)](#).

##### `_M_prior`

`_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_prior` [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M](#) and [\\_M\\_unlink\(\)](#).

##### `_M_sequence`

`const _Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence` [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, m](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator](#)

##### `_M_version`

`unsigned int __gnu_debug::Safe_iterator_base::_M_version` [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_sequence](#)

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_local\\_iterator.h](#)
- [safe\\_local\\_iterator.tcc](#)

### 6.153 `__gnu_debug::Safe_local_iterator_base` Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for `__gnu_debug::_Safe_local_iterator_base`:



### Public Member Functions

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_compare (const _Safe_iterator_base & __x) const noexcept`
- `void _M_invalidate ()`

- void `_M_reset` () noexcept
- const `_Safe_unordered_container_base` \* `_M_safe_container` () const noexcept
- bool `_M_singular` () const noexcept
- void `_M_unlink` () noexcept

### Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- const `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

### Protected Member Functions

- `_Safe_local_iterator_base` ()
- `_Safe_local_iterator_base` (const `_Safe_local_iterator_base` & \_\_x, bool \_\_constant)
- `_Safe_local_iterator_base` (const `_Safe_unordered_container_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant)
- void `_M_attach` (const `_Safe_unordered_container_base` \* \_\_cont, bool \_\_constant)
- void `_M_attach_single` (const `_Safe_sequence_base` \* \_\_seq, bool \_\_constant) noexcept
- void `_M_attach_single` (const `_Safe_unordered_container_base` \* \_\_cont, bool \_\_constant) noexcept
- void `_M_detach` ()
- void `_M_detach_single` () noexcept
- `__gnu_cxx::__mutex` & `_M_get_mutex` () noexcept

#### 6.153.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

#### 6.153.2 Constructor & Destructor Documentation

##### `_Safe_local_iterator_base`() [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base () [inline], [protected]
```

Initializes the iterator and makes it singular.

Referenced by `_Safe_local_iterator_base`().

##### `_Safe_local_iterator_base`() [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
 const _Safe_unordered_container_base * __seq,
 bool __constant) [inline], [protected]
```

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

References `_M_attach`().

**\_Safe\_local\_iterator\_base()** [3/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
 const _Safe_local_iterator_base & __x,
 bool __constant) [inline], [protected]
```

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

References [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_M\\_attach\(\)](#).

**6.153.3 Member Function Documentation****\_M\_attach()** [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References [\\_M\\_attach\(\)](#).

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\(\)](#), and [\\_M\\_attach\(\)](#).

**\_M\_attach()** [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
 const _Safe_unordered_container_base * __cont,
 bool __constant) [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by [\\_Safe\\_local\\_iterator\\_base\(\)](#), [\\_Safe\\_local\\_iterator\\_base\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::\\_M\\_attach\(\)](#).

**\_M\_attach\_single()** [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 const _Safe_sequence_base * __seq,
 bool __constant) [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References [\\_M\\_attach\\_single\(\)](#).

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_M\\_attach\\_single\(\)](#), and [\\_M\\_attach\\_single\(\)](#).

**\_M\_attach\_single()** [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
 const _Safe_unordered_container_base * __cont,
 bool __constant) [protected], [noexcept]
```

Likewise, but not thread-safe.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::\\_M\\_attach\\_single\(\)](#).

**\_M\_attached\_to()**

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

References [\\_M\\_attached\\_to\(\)](#), and [\\_M\\_sequence](#).

Referenced by [\\_M\\_attached\\_to\(\)](#).





### **`_M_unlink()`**

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References [\\_M\\_next](#), [\\_M\\_prior](#), and [\\_M\\_unlink\(\)](#).

Referenced by [\\_M\\_unlink\(\)](#).

## **6.153.4 Member Data Documentation**

### **`_M_next`**

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M](#) and [\\_M\\_unlink\(\)](#).

### **`_M_prior`**

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M](#) and [\\_M\\_unlink\(\)](#).

### **`_M_sequence`**

```
const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, m](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::\\_Safe\\_iterator\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_attached\\_to\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator](#)

### **`_M_version`**

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Referenced by [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_Safe\\_iterator\\_base\(\)](#), [\\_M\\_invalidate\(\)](#), and [\\_\\_gnu\\_debug::\\_Safe\\_sequence](#)

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

## **6.154 `__gnu_debug::_Safe_node_sequence<_Sequence>` Class Template Reference**

```
#include <safe_sequence.h>
```

Inheritance diagram for `__gnu_debug::_Safe_node_sequence<_Sequence>`:



### Public Member Functions

- `_Safe_node_sequence` (`_Safe_node_sequence &&`)=default
- `_Safe_node_sequence` (`_Safe_node_sequence const &`)=default
- `template<typename _Predicate>`  
`constexpr void _M_invalidate_if` (`_Predicate __pred`) const
- `template<typename _Predicate>`  
`void _M_transfer_from_if` (`const _Safe_sequence &__from`, `_Predicate __pred`) const
- `constexpr _Safe_node_sequence & operator=` (`_Safe_node_sequence &&__x`) noexcept
- `constexpr _Safe_node_sequence & operator=` (`const _Safe_node_sequence &`) noexcept

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all` () const
- `void _M_detach_singular` () const
- `__gnu_cxx::__mutex & _M_get_mutex` () const noexcept
- `constexpr void _M_invalidate_all` () const
- `void _M_revalidate_singular` () const
- `void _M_swap` (`const _Safe_sequence_base &__x`) const noexcept

### 6.154.1 Detailed Description

```
template<typename _Sequence>
class __gnu_debug::_Safe_node_sequence< _Sequence >
```

Like `_Safe_sequence` but with a special `_M_invalidate_all` implementation not invalidating past-the-end iterators. Used by node based sequence.

### 6.154.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_sequence\\_base\(\)](#).

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence< \\_Sequence >::\\_M\\_transfer\\_from\\_if\(\)](#).

#### `_M_invalidate_if()`

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
 _Predicate __pred) const [inline], [constexpr], [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**`_M_transfer_from_if()`**

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if (
 const Safe_sequence<_Sequence> & __from,
 _Predicate __pred) const [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

References `std::addressof()`, `__gnu_debug::Safe_sequence_base::M_const_iterators`, `__gnu_debug::Safe_sequence_base::M_g`, `__gnu_debug::Safe_sequence_base::M_iterators`, `__gnu_debug::Safe_iterator_base::M_next`, `__gnu_debug::Safe_iterator_base::M_sequence`, `__gnu_debug::Safe_iterator_base::M_version`, and `__gnu_debug::Safe_sequence`.

**6.154.3 Member Data Documentation****`_M_const_iterators`**

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**`_M_iterators`**

```
Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**`_M_version`**

```
unsigned int __gnu_debug::Safe_sequence_base::M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::Safe_iterator_base::operator=()`, `__gnu_debug::Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

**6.155 `__gnu_debug::Safe_sequence<_Sequence>` Class Template Reference**

```
#include <safe_sequence.h>
```

Inheritance diagram for `__gnu_debug::_Safe_sequence<_Sequence>`:



### Public Member Functions

- `template<typename _Predicate>`  
`constexpr void _M_invalidate_if (_Predicate __pred) const`
- `template<typename _Predicate>`  
`void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const`

### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

### Protected Member Functions

- `void _M_detach_all () const`
- `void _M_detach_singular () const`
- `__gnu_cxx::__mutex & _M_get_mutex () const noexcept`
- `constexpr void _M_invalidate_all () const`
- `void _M_revalidate_singular () const`
- `void _M_swap (const _Safe_sequence_base &__x) const noexcept`

#### 6.155.1 Detailed Description

```

template<typename _Sequence>
class __gnu_debug::_Safe_sequence<_Sequence>

```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

### 6.155.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_sequence\\_base\(\)](#).

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

#### `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [constexpr], [protected], [inherited]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

#### `_M_invalidate_if()`

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (
 _Predicate __pred) const [inline], [constexpr]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).





- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int M version`

## Protected Member Functions

- constexpr **\_Safe\_sequence\_base** (\_Safe\_sequence\_base &&\_\_seq) noexcept
- constexpr **\_Safe\_sequence\_base** (const \_Safe\_sequence\_base &) noexcept
- constexpr **~\_Safe\_sequence\_base** () noexcept
- void **\_M\_detach\_all** () const
- void **\_M\_detach\_singular** () const
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () const noexcept
- constexpr void **\_M\_invalidate\_all** () const
- void **\_M\_revalidate\_singular** () const
- void **\_M\_swap** (const \_Safe\_sequence\_base &\_\_x) const noexcept
- constexpr \_Safe\_sequence\_base & **operator=** (\_Safe\_sequence\_base &&\_\_x) noexcept
- constexpr \_Safe\_sequence\_base & **operator=** (const \_Safe\_sequence\_base &) noexcept

## Friends

- class **\_Safe\_iterator\_base**

### 6.156.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

### 6.156.2 Constructor & Destructor Documentation

#### `~_Safe_sequence_base()`

```
__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base () [inline], [constexpr], [protected],
[noexcept]
```

Notify all iterators that reference this sequence that the sequence is being destroyed.

References `_M_detach_all()`.

### 6.156.3 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected]
```

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

**`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::Safe_sequence_base::_M_get_mutex () const [protected], [noexcept]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_invalidate_all()`**

```
void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline], [constexpr], [protected]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

**`_M_revalidate_singular()`**

```
void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () const [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**`_M_swap()`**

```
void __gnu_debug::Safe_sequence_base::_M_swap (
 const Safe_sequence_base & __x) const [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**6.156.4 Member Data Documentation****`_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [mutable]
```

The list of constant iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [mutable]
```

The list of mutable iterators that reference this container.

Referenced by [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

**`_M_version`**

```
unsigned int __gnu_debug::Safe_sequence_base::_M_version [mutable]
```

The container version number. This number may never be 0.

Referenced by [\\_M\\_invalidate\\_all\(\)](#), [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::operator=\(\)](#), and [\\_\\_gnu\\_debug::Safe\\_local\\_iterator<\\_OtherIterator, UContainer>::operator=\(\)](#).

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

**6.157 `__gnu_debug::Safe_unordered_container<_Container>` Class Template Reference**

```
#include <safe_unordered_container.h>
```

Inheritance diagram for `__gnu_debug::__Safe_unordered_container<_Container>`:



## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all () const`
- `void _M_detach_singular () const`
- `__gnu_cxx::__mutex & _M_get_mutex () const noexcept`
- `template<typename _VictimIt>  
void _M_invalidate (_VictimIt __victim)`
- `template<typename _VictimIt>  
void _M_invalidate (_VictimIt __victim, const __gnu_cxx::__scoped_lock &__lock)`
- `constexpr void _M_invalidate_all () const`
- `void _M_invalidate_all ()`
- `template<typename _Predicate>  
void _M_invalidate_all_if (_Predicate __pred)`
- `void _M_revalidate_singular () const`
- `void _M_swap (const _Safe_sequence_base &__x) const noexcept`
- `void _M_swap (const _Safe_unordered_container_base &__x) const noexcept`

### 6.157.1 Detailed Description

```
template<typename _Container>
class __gnu_debug::__Safe_unordered_container<_Container>
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their

operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

### 6.157.2 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by [~\\_Safe\\_unordered\\_container\\_base\(\)](#).

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### `_M_get_mutex()`

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
```

For use in `_Safe_sequence`.

Referenced by [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#).

#### `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [constexpr], [protected], [inherited]
```

Invalidates all iterators.

References [\\_M\\_invalidate\\_all\(\)](#), and [\\_M\\_version](#).

Referenced by [\\_M\\_invalidate\\_all\(\)](#).

#### `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

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

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 const _Safe_unordered_container_base & __x) const [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 6.157.3 Member Data Documentation

#### **`_M_const_iterators`**

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [mutable], [inherited]

The list of constant iterators that reference this container.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

#### **`_M_const_local_iterators`**

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators` [mutable], [inherited]

The list of constant local iterators that reference this container.

#### **`_M_iterators`**

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [mutable], [inherited]

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`.

#### **`_M_local_iterators`**

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [mutable], [inherited]

The list of mutable local iterators that reference this container.

#### **`_M_version`**

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Referenced by `_M_invalidate_all()`, `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator<_Base_iterator, map>::operator=()`, `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`, and `__gnu_debug::_Safe_local_iterator<_OtherIterator, _UContainer>::operator=()`.

The documentation for this class was generated from the following files:

- `safe_unordered_container.h`
- `safe_unordered_container.tcc`

### 6.158 `__gnu_debug::_Safe_unordered_container_base` Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for \_\_gnu\_debug::\_\_Safe\_unordered\_container\_base:



## Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_const_local_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- `_Safe_iterator_base` \* `_M_local_iterators`
- `unsigned int` `_M_version`

## Protected Member Functions

- `_Safe_unordered_container_base` (`_Safe_unordered_container_base` &&\_\_x) noexcept

- **\_Safe\_unordered\_container\_base** (const \_Safe\_unordered\_container\_base &) noexcept
- **~\_Safe\_unordered\_container\_base** () noexcept
- void **\_M\_detach\_all** () const
- void **\_M\_detach\_singular** () const
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () const noexcept
- constexpr void **\_M\_invalidate\_all** () const
- void **\_M\_revalidate\_singular** () const
- void **\_M\_swap** (const \_Safe\_sequence\_base &\_\_x) const noexcept
- void **\_M\_swap** (const \_Safe\_unordered\_container\_base &\_\_x) const noexcept
- \_Safe\_unordered\_container\_base & **operator=** (\_Safe\_unordered\_container\_base &&)=default
- \_Safe\_unordered\_container\_base & **operator=** (\_Safe\_unordered\_container\_base const &)=default

## Friends

- class **\_Safe\_local\_iterator\_base**

### 6.158.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

### 6.158.2 Constructor & Destructor Documentation

#### `~_Safe_unordered_container_base()`

```
__gnu_debug::_Safe_unordered_container_base::~~_Safe_unordered_container_base () [inline], [protected], [noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.

References `_M_detach_all()`.

### 6.158.3 Member Function Documentation

#### `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const [protected]
```

Detach all iterators, leaving them singular.

Referenced by `~_Safe_unordered_container_base()`.

#### `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.



**`_M_get_mutex()`**

```
__gnu_cxx::__mutex & __gnu_debug::Safe_sequence_base::_M_get_mutex () const [protected], [noexcept],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**`_M_invalidate_all()`**

```
void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline], [constexpr], [protected],
[inherited]
```

Invalidates all iterators.

References `_M_invalidate_all()`, and `_M_version`.

Referenced by `_M_invalidate_all()`.

**`_M_revalidate_singular()`**

```
void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

**`_M_swap()` [1/2]**

```
void __gnu_debug::Safe_sequence_base::_M_swap (
 const Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**`_M_swap()` [2/2]**

```
void __gnu_debug::Safe_unordered_container_base::_M_swap (
 const Safe_unordered_container_base & __x) const [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

**6.158.4 Member Data Documentation****`_M_const_iterators`**

```
_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**`_M_const_local_iterators`**

```
_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::_M_const_local_iterators [mutable]
```

The list of constant local iterators that reference this container.

**`_M_iterators`**

```
_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if()`.

**\_M\_local\_iterators**

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [mutable]
```

The list of mutable local iterators that reference this container.

**\_M\_version**

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by [\\_M\\_invalidate\\_all\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>::\\_M\\_transfer\\_from\\_if\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Base\\_iterator, map>::operator=\(\)](#), [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>](#) and [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator<\\_OtherIterator, \\_UContainer>::operator=\(\)](#).

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

### 6.159 **\_\_gnu\_debug::\_Safe\_vector<\_SafeSequence, \_BaseSequence> Class Template Reference**

```
#include <vector>
```

Inheritance diagram for `__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence>`:

**Protected Member Functions**

- constexpr **\_Safe\_vector** (`_Safe_vector &&__x`) noexcept
- constexpr **\_Safe\_vector** (`const _Safe_vector &`) noexcept
- constexpr **\_Safe\_vector** (`size_type __n`) noexcept
- bool **\_M\_requires\_reallocation** (`size_type __elements`) const noexcept
- constexpr void **\_M\_update\_guaranteed\_capacity** () noexcept

- constexpr `_Safe_vector & operator=` (`_Safe_vector &&__x`) noexcept
- constexpr `_Safe_vector & operator=` (`const _Safe_vector &`) noexcept

### Protected Attributes

- `size_type _M_guaranteed_capacity`

#### 6.159.1 Detailed Description

```
template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug::__Safe_vector< _SafeSequence, _BaseSequence >
```

Base class for Debug Mode vector.

Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.

The documentation for this class was generated from the following file:

- [debug/vector](#)

## 6.160 `__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- void `operator()` (`_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp`)

#### 6.160.1 Detailed Description

```
template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
```

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.161 `__gnu_parallel::__SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

```
#include <multiway_merge.h>
```

### Public Member Functions

- void `operator()` (`_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp`)

#### 6.161.1 Detailed Description

```
template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::__SamplingSorter< false, _RAIter, _StrictWeakOrdering >
```

Non-`__stable` sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

## 6.162 std::\_\_detail::\_Scanner<\_CharT> Class Template Reference

```
#include <regex_scanner.h>
```

### Public Types

- typedef const [std::ctype](#)<\_CharT> **\_CtypeT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef [std::basic\\_string](#)<\_CharT> **\_StringT**
- enum **\_TokenT** : unsigned {  
     **\_S\_token\_anychar** , **\_S\_token\_ord\_char** , **\_S\_token\_oct\_num** , **\_S\_token\_hex\_num** ,  
     **\_S\_token\_backref** , **\_S\_token\_subexpr\_begin** , **\_S\_token\_subexpr\_no\_group\_begin** , **\_S\_token\_↵**  
     **subexpr\_lookahead\_begin** ,  
     **\_S\_token\_subexpr\_end** , **\_S\_token\_bracket\_begin** , **\_S\_token\_bracket\_neg\_begin** , **\_S\_token\_bracket\_↵**  
     **end** ,  
     **\_S\_token\_interval\_begin** , **\_S\_token\_interval\_end** , **\_S\_token\_quoted\_class** , **\_S\_token\_char\_class\_name**  
     ,  
     **\_S\_token\_collsymbol** , **\_S\_token\_equiv\_class\_name** , **\_S\_token\_opt** , **\_S\_token\_or** ,  
     **\_S\_token\_closure0** , **\_S\_token\_closure1** , **\_S\_token\_line\_begin** , **\_S\_token\_line\_end** ,  
     **\_S\_token\_word\_bound** , **\_S\_token\_comma** , **\_S\_token\_dup\_count** , **\_S\_token\_eof** ,  
     **\_S\_token\_bracket\_dash** , **\_S\_token\_unknown** }

### Public Member Functions

- **\_Scanner** (const \_CharT \*\_\_begin, const \_CharT \*\_\_end, [\\_FlagT](#) \_\_flags, [std::locale](#) \_\_loc)
- void **\_M\_advance** ()
- **\_TokenT \_M\_get\_token** () const noexcept
- const **\_StringT &\_M\_get\_value** () const noexcept

### Protected Types

- enum **\_StateT** { **\_S\_state\_normal** , **\_S\_state\_in\_brace** , **\_S\_state\_in\_bracket** }

### Protected Member Functions

- const char \* **\_M\_find\_escape** (char \_\_c)
- bool **\_M\_is\_awk** () const
- bool **\_M\_is\_basic** () const
- bool **\_M\_is\_ecma** () const
- bool **\_M\_is\_extended** () const
- bool **\_M\_is\_grep** () const

### Protected Attributes

- bool **\_M\_at\_bracket\_start**
- const [std::pair](#)< char, char > **\_M\_awk\_escape\_tbl** [11]
- const char \* **\_M\_basic\_spec\_char**
- const [std::pair](#)< char, char > **\_M\_ecma\_escape\_tbl** [8]
- const char \* **\_M\_ecma\_spec\_char**
- const [std::pair](#)< char, char > \* **\_M\_escape\_tbl**
- const char \* **\_M\_extended\_spec\_char**
- [\\_FlagT](#) **\_M\_flags**
- const char \* **\_M\_spec\_char**
- **\_StateT** **\_M\_state**
- **\_TokenT** **\_M\_token**
- const [std::pair](#)< char, **\_TokenT** > **\_M\_token\_tbl** [9]

### 6.162.1 Detailed Description

```
template<typename _CharT>
class std::__detail::_Scanner<_CharT>
```

Scans an input range for regex tokens.

The `_Scanner` class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

### 6.162.2 Member Enumeration Documentation

#### `_TokenT`

```
enum std::__detail::_ScannerBase::_TokenT : unsigned [inherited]
```

Token types returned from the scanner.

The documentation for this class was generated from the following files:

- [regex\\_scanner.h](#)
- [regex\\_scanner.tcc](#)

## 6.163 `__gnu_debug::_Sequence_traits<_Sequence>` Struct Template Reference

```
#include <safe_iterator.h>
```

### Public Types

- typedef `_Distance_traits< typename _Sequence::iterator >_DistTraits`

### Static Public Member Functions

- static `_DistTraits::__type _S_size (const _Sequence &__seq)`

### 6.163.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_Sequence_traits<_Sequence>
```

Sequence traits giving the size of a container if possible.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 6.164 `__gnu_parallel::_Settings` Struct Reference

```
#include <settings.h>
```

### Static Public Member Functions

- static const `_Settings & get ()` throw ()
- static void [set](#) (`_Settings &`) throw ()

**Public Attributes**

- [\\_SequenceIndex](#) `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- [\\_AlgorithmStrategy](#) `algorithm_strategy`
- unsigned int `cache_line_size`
- [\\_SequenceIndex](#) `count_minimal_n`
- [\\_SequenceIndex](#) `fill_minimal_n`
- [\\_FindAlgorithm](#) `find_algorithm`
- double `find_increasing_factor`
- [\\_SequenceIndex](#) `find_initial_block_size`
- [\\_SequenceIndex](#) `find_maximum_block_size`
- float `find_scale_factor`
- [\\_SequenceIndex](#) `find_sequential_search_size`
- [\\_SequenceIndex](#) `for_each_minimal_n`
- [\\_SequenceIndex](#) `generate_minimal_n`
- unsigned long long `L1_cache_size`
- unsigned long long `L2_cache_size`
- [\\_SequenceIndex](#) `max_element_minimal_n`
- [\\_SequenceIndex](#) `merge_minimal_n`
- unsigned int `merge_oversampling`
- [\\_SplittingAlgorithm](#) `merge_splitting`
- [\\_SequenceIndex](#) `min_element_minimal_n`
- [\\_MultiwayMergeAlgorithm](#) `multiway_merge_algorithm`
- int `multiway_merge_minimal_k`
- [\\_SequenceIndex](#) `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- [\\_SplittingAlgorithm](#) `multiway_merge_splitting`
- [\\_SequenceIndex](#) `nth_element_minimal_n`
- [\\_SequenceIndex](#) `partial_sort_minimal_n`
- [\\_PartialSumAlgorithm](#) `partial_sum_algorithm`
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- [\\_SequenceIndex](#) `partition_chunk_size`
- [\\_SequenceIndex](#) `partition_minimal_n`
- [\\_SequenceIndex](#) `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- [\\_SequenceIndex](#) `replace_minimal_n`
- [\\_SequenceIndex](#) `search_minimal_n`
- [\\_SequenceIndex](#) `set_difference_minimal_n`
- [\\_SequenceIndex](#) `set_intersection_minimal_n`
- [\\_SequenceIndex](#) `set_symmetric_difference_minimal_n`
- [\\_SequenceIndex](#) `set_union_minimal_n`
- [\\_SortAlgorithm](#) `sort_algorithm`
- [\\_SequenceIndex](#) `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`
- unsigned int `sort_qs_num_samples_preset`
- [\\_SequenceIndex](#) `sort_qsb_base_case_maximal_n`
- [\\_SplittingAlgorithm](#) `sort_splitting`
- unsigned int `TLB_size`
- [\\_SequenceIndex](#) `transform_minimal_n`
- [\\_SequenceIndex](#) `unique_copy_minimal_n`
- [\\_SequenceIndex](#) `workstealing_chunk_size`

### 6.164.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

### 6.164.2 Member Function Documentation

#### `get()`

```
const _Settings & __gnu_parallel::_Settings::get () throw () [static]
```

Get the global settings.

Referenced by `__gnu_parallel::_find_template()`, `__gnu_parallel::_find_template()`, `__gnu_parallel::_find_template()`, `__gnu_parallel::_for_each_template_random_access_workstealing()`, `__gnu_parallel::_parallel_nth_element()`, `__gnu_parallel::_parallel_partial_sum()`, `__gnu_parallel::_parallel_partial_sum_linear()`, `__gnu_parallel::_parallel_partition()`, `__gnu_parallel::_parallel_random_shuffle_drs()`, `__gnu_parallel::_parallel_sort()`, `__gnu_parallel::_parallel_sort()`, `__gnu_parallel::_parallel_sort_qs_conquer()`, `__gnu_parallel::_qsb_local_sort_with_helping()`, `__gnu_parallel::_sequential_random_s`, `__gnu_parallel::multiway_merge_sampling_splitting()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### `set()`

```
void __gnu_parallel::_Settings::set (
 _Settings &) throw () [static]
```

Set the global settings.

References `accumulate_minimal_n`, `adjacent_difference_minimal_n`, `cache_line_size`, `count_minimal_n`, `fill_minimal_n`, `find_increasing_factor`, `find_initial_block_size`, `find_maximum_block_size`, `find_scale_factor`, `find_sequential_search_size`, `for_each_minimal_n`, `generate_minimal_n`, `L1_cache_size`, `L2_cache_size`, `max_element_minimal_n`, `merge_minimal_n`, `merge_oversampling`, `min_element_minimal_n`, `multiway_merge_minimal_k`, `multiway_merge_minimal_n`, `multiway_merge_oversampling`, `nth_element_minimal_n`, `partial_sort_minimal_n`, `partial_sum_dilation`, `partial_sum_minimal_n`, `partition_chunk_share`, `partition_chunk_size`, `partition_minimal_n`, `qsb_steals`, `random_shuffle_minimal_n`, `replace_minimal_n`, `search_minimal_n`, `set()`, `set_difference_minimal_n`, `set_intersection_minimal_n`, `set_symmetric_difference_minimal_n`, `set_union_minimal_n`, `sort_minimal_n`, `sort_mwms_oversampling`, `sort_qs_num_samples_preset`, `sort_qsb_base_case_maximal_n`, `TLB_size`, `transform_minimal_n`, and `unique_copy_minimal_n`.

Referenced by `set()`.

### 6.164.3 Member Data Documentation

#### `accumulate_minimal_n`

```
_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n
```

Minimal input size for accumulate.

Referenced by `set()`.

#### `adjacent_difference_minimal_n`

```
unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n
```

Minimal input size for adjacent\_difference.

Referenced by `set()`.

#### `cache_line_size`

```
unsigned int __gnu_parallel::_Settings::cache_line_size
```

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`, and `set()`.

**count\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::count_minimal_n`

Minimal input size for count and count\_if.

Referenced by [set\(\)](#).

**fill\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::fill_minimal_n`

Minimal input size for fill.

Referenced by [set\(\)](#).

**find\_increasing\_factor**

`double __gnu_parallel::_Settings::find_increasing_factor`

Block size increase factor for find.

Referenced by [set\(\)](#).

**find\_initial\_block\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::find_initial_block_size`

Initial block size for find.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**find\_maximum\_block\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::find_maximum_block_size`

Maximal block size for find.

Referenced by [set\(\)](#).

**find\_scale\_factor**

`float __gnu_parallel::_Settings::find_scale_factor`

Block size scale-down factor with respect to current position.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**find\_sequential\_search\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::find_sequential_search_size`

Start with looking for this many elements sequentially, for find.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), and [set\(\)](#).

**for\_each\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::for_each_minimal_n`

Minimal input size for for\_each.

Referenced by [set\(\)](#).

**generate\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for generate.

Referenced by [set\(\)](#).



**L1\_cache\_size**

`unsigned long long __gnu_parallel::_Settings::L1_cache_size`  
size of the L1 cache in bytes (underestimation).

Referenced by [set\(\)](#).

**L2\_cache\_size**

`unsigned long long __gnu_parallel::_Settings::L2_cache_size`  
size of the L2 cache in bytes (underestimation).

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

**max\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::max_element_minimal_n`  
Minimal input size for `max_element`.

Referenced by [set\(\)](#).

**merge\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::merge_minimal_n`  
Minimal input size for `merge`.

Referenced by [set\(\)](#).

**merge\_oversampling**

`unsigned int __gnu_parallel::_Settings::merge_oversampling`  
Oversampling factor for `merge`.

Referenced by [\\_\\_gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#), and [set\(\)](#).

**min\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::min_element_minimal_n`  
Minimal input size for `min_element`.

Referenced by [set\(\)](#).

**multiway\_merge\_minimal\_k**

`int __gnu_parallel::_Settings::multiway_merge_minimal_k`  
Oversampling factor for `multiway_merge`.

Referenced by [set\(\)](#).

**multiway\_merge\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::multiway_merge_minimal_n`  
Minimal input size for `multiway_merge`.

Referenced by [set\(\)](#).

**multiway\_merge\_oversampling**

`unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`  
Oversampling factor for `multiway_merge`.

Referenced by [set\(\)](#).

**nth\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::nth_element_minimal_n`

Minimal input size for `nth_element`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element\(\)](#), and [set\(\)](#).

**partial\_sort\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partial_sort_minimal_n`

Minimal input size for `partial_sort`.

Referenced by [set\(\)](#).

**partial\_sum\_dilation**

`float __gnu_parallel::_Settings::partial_sum_dilation`

Ratio for `partial_sum`. Assume "sum and write result" to be this factor slower than just "sum".

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partial\\_sum\\_linear\(\)](#), and [set\(\)](#).

**partial\_sum\_minimal\_n**

`unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`

Minimal input size for `partial_sum`.

Referenced by [set\(\)](#).

**partition\_chunk\_share**

`double __gnu_parallel::_Settings::partition_chunk_share`

Chunk size for partition, relative to input size. If  $> 0.0$ , this value overrides `partition_chunk_size`.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#), and [set\(\)](#).

**partition\_chunk\_size**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partition_chunk_size`

Chunk size for partition.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#), and [set\(\)](#).

**partition\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::partition_minimal_n`

Minimal input size for partition.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element\(\)](#), and [set\(\)](#).

**qsb\_steals**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::qsb_steals`

The number of stolen ranges in load-balanced quicksort.

Referenced by [set\(\)](#).

**random\_shuffle\_minimal\_n**

`unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n`

Minimal input size for `random_shuffle`.

Referenced by [set\(\)](#).

**replace\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Referenced by [set\(\)](#).

**search\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Referenced by [set\(\)](#).

**set\_difference\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Referenced by [set\(\)](#).

**set\_intersection\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Referenced by [set\(\)](#).

**set\_symmetric\_difference\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Referenced by [set\(\)](#).

**set\_union\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Referenced by [set\(\)](#).

**sort\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Referenced by [set\(\)](#).

**sort\_mwms\_oversampling**

`unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for `parallel_std::sort` (MWMS).

Referenced by [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\(\)](#), [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu\(\)](#), and [set\(\)](#).

**sort\_qs\_num\_samples\_preset**

`unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Referenced by [set\(\)](#).

**sort\_qsb\_base\_case\_maximal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping\(\)](#), and [set\(\)](#).

**TLB\_size**

`unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

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

**transform\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Referenced by [set\(\)](#).

**unique\_copy\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Referenced by [set\(\)](#).

The documentation for this struct was generated from the following file:

- [settings.h](#)

**6.165 `std::_Sp_ebo_helper<_Tp, false >` Struct Template Reference**

```
#include <shared_ptr_base.h>
```

**Public Attributes**

- `_Tp _M_obj`

**6.165.1 Detailed Description**

```
template<typename _Tp>
```

```
struct std::_Sp_ebo_helper<_Tp, false >
```

Specialization not using `[[no_unique_address]]`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

**6.166 `std::_Sp_ebo_helper<_Tp, true >` Struct Template Reference**

```
#include <shared_ptr_base.h>
```

**Public Attributes**

- `_Tp _M_obj`

### 6.166.1 Detailed Description

```
template<typename _Tp>
struct std::_Sp_ebo_helper< _Tp, true >
```

Specialization using `[[no_unique_address]]`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 6.167 `__gnu_parallel::_SplitConsistently< __exact, _RAlter, _Compare, _SortingPlacesIterator > Struct` Template Reference

```
#include <multiway_mergesort.h>
```

### 6.167.1 Detailed Description

```
template<bool __exact, typename _RAlter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAlter, _Compare, _SortingPlacesIterator >
```

Split consistently.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 6.168 `__gnu_parallel::_SplitConsistently< false, _RAlter, _Compare, _SortingPlacesIterator > Struct` Template Reference

```
#include <multiway_mergesort.h>
```

### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWSSortingData](#)< \_RAlter > \*\_\_sd, \_Compare &\_\_comp, const typename [std::iterator\\_traits](#)< \_RAlter >::difference\_type \_\_num\_samples) const

### 6.168.1 Detailed Description

```
template<typename _RAlter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAlter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 6.169 `__gnu_parallel::_SplitConsistently< true, _RAlter, _Compare, _SortingPlacesIterator > Struct` Template Reference

```
#include <multiway_mergesort.h>
```

### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWSSortingData](#)< \_RAlter > \*\_\_sd, \_Compare &\_\_comp, const typename [std::iterator\\_traits](#)< \_RAlter >::difference\_type \_\_num\_samples) const

### 6.169.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::__SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 6.170 std::\_\_detail::\_\_StateSeq< \_TraitsT > Class Template Reference

```
#include <regex_automaton.h>
```

### Public Types

- typedef \_NFA< \_TraitsT > \_RegexT

### Public Member Functions

- **\_StateSeq** (\_RegexT &\_\_nfa, \_StateIdT \_\_s)
- **\_StateSeq** (\_RegexT &\_\_nfa, \_StateIdT \_\_s, \_StateIdT \_\_end)
- void **\_M\_append** (\_StateIdT \_\_id)
- void **\_M\_append** (const \_StateSeq &\_\_s)
- **\_StateSeq** **\_M\_clone** ()

### Public Attributes

- \_StateIdT **\_M\_end**
- \_RegexT & **\_M\_nfa**
- \_StateIdT **\_M\_start**

### 6.170.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::__StateSeq< _TraitsT >
```

Describes a sequence of one or more \_State, its current start and end(s). This structure contains fragments of an NFA during construction.

The documentation for this class was generated from the following files:

- [regex\\_automaton.h](#)
- [regex\\_automaton.tcc](#)

## 6.171 \_\_gnu\_cxx::\_\_Std\_pointer\_impl< \_Tp > Class Template Reference

```
#include <pointer.h>
```

### Public Types

- typedef \_Tp **element\_type**

### Public Member Functions

- \_Tp \* **get** () const
- bool **operator**< (const [\\_Std\\_pointer\\_impl](#) &\_\_arg) const
- bool **operator**== (const [\\_Std\\_pointer\\_impl](#) &\_\_arg) const
- void **set** (element\_type \*\_\_arg)

### 6.171.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::Std_pointer_impl< _Tp >
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 6.172 \_\_gnu\_cxx::Temporary\_buffer<\_ForwardIterator, \_Tp> Class Template Reference

```
#include <stl_tempbuf.h>
```

Inheritance diagram for `__gnu_cxx::Temporary_buffer<_ForwardIterator, _Tp>`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- `_Temporary_buffer` (`_ForwardIterator __seed`, `size_type __original_len`)
- `size_type _M_requested_size` () const
- iterator `begin` ()
- iterator `end` ()
- `size_type size` () const

## Protected Attributes

- struct std::\_Temporary\_buffer::\_Impl **\_M\_impl**
- size\_type **\_M\_original\_len**

### 6.172.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

### 6.172.2 Constructor & Destructor Documentation

#### `_Temporary_buffer()`

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
 _ForwardIterator __seed,
 size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

### 6.172.3 Member Function Documentation

#### `_M_requested_size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const [inline]
```

Returns the size requested by the constructor; may be `>size()`.

#### `begin()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline]
```

As per Table mumble.

#### `end()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline]
```

As per Table mumble.

#### `size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]
```

As per Table mumble.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

## 6.173 `std::_Temporary_buffer< _ForwardIterator, _Tp >` Class Template Reference

```
#include <stl_tempbuf.h>
```



Inheritance diagram for std::\_Temporary\_buffer<\_ForwardIterator, \_Tp>:



### Public Types

- typedef pointer **iterator**
- typedef `value_type * pointer`
- typedef `ptrdiff_t size_type`
- typedef `_Tp value_type`

### Public Member Functions

- `_Temporary_buffer` (`_ForwardIterator __seed`, `size_type __original_len`)
- `size_type M_requested_size` () const
- `iterator begin` ()
- `iterator end` ()
- `size_type size` () const

### Protected Attributes

- struct `std::_Temporary_buffer::Impl` **M\_impl**
- `size_type` **M\_original\_len**

#### 6.173.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

#### 6.173.2 Constructor & Destructor Documentation

##### `_Temporary_buffer()`

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer (
```

```
 _ForwardIterator __seed,
 size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

References [begin\(\)](#), and [end\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp>::temporary\\_buffer\(\)](#).

### 6.173.3 Member Function Documentation

#### **`_M_requested_size()`**

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer<_ForwardIterator, _Tp>::_M_requested_size () const [inline]
Returns the size requested by the constructor; may be >size().
```

#### **`begin()`**

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::begin () [inline]
As per Table mumble.
Referenced by _Temporary_buffer\(\).
```

#### **`end()`**

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::end () [inline]
As per Table mumble.
Referenced by _Temporary_buffer\(\).
```

#### **`size()`**

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer<_ForwardIterator, _Tp>::size () const [inline]
As per Table mumble.
The documentation for this class was generated from the following file:
```

- [stl\\_tempbuf.h](#)

## 6.174 `__gnu_cxx::Unqualified_type<_Tp>` Struct Template Reference

```
#include <pointer.h>
```

### Public Types

- `typedef _Tp type`

#### 6.174.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::Unqualified_type<_Tp>
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_of` is still `T`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

## 6.175 std::\_Vector\_base&lt;\_Tp, \_Alloc&gt; Struct Template Reference

```
#include <stl_vector.h>
```

Inheritance diagram for std::\_Vector\_base<\_Tp, \_Alloc>:



## Public Types

- typedef `__gnu_cxx::__alloc_traits<_Alloc>::template rebind<_Tp>::other` **\_Tp\_alloc\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__alloc_traits<_Tp_alloc_type>::pointer` **pointer**

## Public Member Functions

- constexpr **\_Vector\_base** (`_Tp_alloc_type &&__a`) noexcept
- **\_Vector\_base** (`_Vector_base &&`)=default
- constexpr **\_Vector\_base** (`_Vector_base &&__x, const allocator_type &__a`)
- constexpr **\_Vector\_base** (`const allocator_type &__a`) noexcept
- constexpr **\_Vector\_base** (`const allocator_type &__a, _Vector_base &&__x`)
- constexpr **\_Vector\_base** (`size_t __n`)
- constexpr **\_Vector\_base** (`size_t __n, const allocator_type &__a`)
- constexpr pointer **\_M\_allocate** (`size_t __n`)
- constexpr `_Alloc_result` **\_M\_allocate\_at\_least** (`size_t __n`)
- constexpr void **\_M\_deallocate** (`pointer __p, size_t __n`)
- constexpr const `_Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr `_Tp_alloc_type &` **\_M\_get\_Tp\_allocator** () noexcept
- constexpr `allocator_type` **get\_allocator** () const noexcept

## Static Public Member Functions

- static constexpr `size_t` **\_S\_max\_size** (`const _Tp_alloc_type &__a`) noexcept

## Public Attributes

- `_Vector_impl` **\_M\_impl**

## Protected Member Functions

- constexpr void **\_M\_create\_storage** (`size_t __n`)

### 6.175.1 Detailed Description

**template<typename \_Tp, typename \_Alloc>**  
**struct std::\_Vector\_base<\_Tp, \_Alloc>**

See `bits/stl_deque.h`'s `_Deque_base` for an explanation.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

### 6.176 std::add\_const<\_Tp> Struct Template Reference

```
#include <type_traits>
```

## Public Types

- using **type**

### 6.176.1 Detailed Description

```
template<typename _Tp>
struct std::add_const< _Tp >
```

add\_const

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.177 std::add\_cv< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.177.1 Detailed Description

```
template<typename _Tp>
struct std::add_cv< _Tp >
```

add\_cv

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.178 std::add\_lvalue\_reference< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.178.1 Detailed Description

```
template<typename _Tp>
struct std::add_lvalue_reference< _Tp >
```

add\_lvalue\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.179 std::add\_pointer< \_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.179.1 Detailed Description

```
template<typename _Tp>
struct std::add_pointer<_Tp >
```

add\_pointer

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.180 std::add\_rvalue\_reference<\_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.180.1 Detailed Description

```
template<typename _Tp>
struct std::add_rvalue_reference<_Tp >
```

add\_rvalue\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.181 std::add\_volatile<\_Tp > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.181.1 Detailed Description

```
template<typename _Tp>
struct std::add_volatile<_Tp >
```

add\_volatile

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.182 std::adopt\_lock\_t Struct Reference

```
#include <std_mutex.h>
```

### 6.182.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 6.183 `std::aligned_storage< _Len, _Align >` Struct Template Reference

```
#include <type_traits>
```

### 6.183.1 Detailed Description

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
struct std::aligned_storage< _Len, _Align >
```

Aligned storage.

The member typedef `type` is be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

It is important to use the nested `type` as uninitialized storage, not the `std::aligned_storage` type itself which is an empty class with 1-byte alignment. So this is correct:

```
typename std::aligned_storage<sizeof(X), alignof(X)>::type m_xobj;
```

This is wrong:

```
std::aligned_storage<sizeof(X), alignof(X)> m_xobj;
```

In C++14 and later `std::aligned_storage_t<sizeof(X), alignof(X)>` can be used to refer to the type member typedef.

The default value of `_Align` is supposed to be the most stringent fundamental alignment requirement for any C++ object type whose size is no greater than `_Len` (see [basic.align] in the C++ standard).

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

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.184 `std::aligned_union< _Len, _Types >` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using [type](#)

### Static Public Attributes

- static const size\_t [alignment\\_value](#)

### 6.184.1 Detailed Description

```
template<size_t _Len, typename... _Types>
struct std::aligned_union< _Len, _Types >
```

Provide aligned storage for types.

[meta.trans.other]

Provides aligned storage for any of the provided types of at least size `_Len`.

See also

[aligned\\_storage](#)

**Deprecated** Deprecated in C++23.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.185 `std::alignment_of< _Tp >` Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for `std::alignment_of< _Tp >`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr `std::size_t` **value**

#### 6.185.1 Detailed Description

```
template<typename _Tp>
```

```
struct std::alignment_of< _Tp >
```

`alignment_of`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.186 `__gnu_cxx::allocator< _Tp >` Class Template Reference

```
#include <memory>
```



Inheritance diagram for `__gnu_cxx::allocator<_Tp>`:



### Public Types

- typedef ptrdiff\_t **difference\_type**
- using **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **allocator** (const allocator &\_\_a) noexcept
- template<typename \_Tp1>  
constexpr **allocator** (const allocator<\_Tp1> &) noexcept
- constexpr \_Tp \* **allocate** (size\_t \_\_n)
- constexpr \_Tp \* **allocate** (size\_type \_\_n, const void \* \_\_p) noexcept
- constexpr allocation\_result<\_Tp \*, size\_t> **allocate\_at\_least** (size\_t \_\_n)
- constexpr void **deallocate** (\_Tp \* \_\_p, size\_t \_\_n)
- allocator & **operator=** (const allocator &) = default

### Static Public Member Functions

- static constexpr void **\_S\_check\_allocation\_limit** (size\_t \_\_n)

### Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_T1, typename \_T2>  
constexpr bool **operator==** (const allocator<\_T1> &, const allocator<\_T2> &) noexcept

### 6.186.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::allocator<_Tp >
```

The *standard* allocator, as per C++03 [20.4.1].

See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator> for further details.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 6.187 std::allocator<\_Tp> Class Template Reference

```
#include <memory>
```

Inheritance diagram for `std::allocator<_Tp>`:



#### Public Types

- typedef ptrdiff\_t **difference\_type**
- using **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **allocator** (const allocator &\_\_a) noexcept
- template<typename \_Tp1>  
constexpr **allocator** (const allocator<\_Tp1> &) noexcept
- constexpr \_Tp \* **allocate** (size\_type \_\_n, const void \* \_\_p = static\_cast<const void \* >(0))
- constexpr \_Tp \* **allocate** (size\_t \_\_n)
- constexpr allocation\_result<\_Tp \*, size\_t> **allocate\_at\_least** (size\_t \_\_n)
- constexpr void **deallocate** (\_Tp \* \_\_p, size\_t \_\_n)
- allocator & **operator=** (const allocator &) = default

### Static Public Member Functions

- static constexpr void **\_S\_check\_allocation\_limit** (size\_t \_\_n)

### Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_T1, typename \_T2>  
constexpr bool **operator==** (const allocator< \_T1 > &, const allocator< \_T2 > &) noexcept

#### 6.187.1 Detailed Description

template<typename \_Tp>  
class std::allocator< \_Tp >

The *standard* allocator, as per C++03 [20.4.1].

See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator> for further details.

### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 6.188 std::allocator< void > Class Reference

```
#include <memory>
```

Inheritance diagram for std::allocator< void >:



### Public Types

- typedef ptrdiff\_t **difference\_type**

- typedef ptrdiff\_t **difference\_type**
- using **propagate\_on\_container\_move\_assignment**
- using **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef size\_t **size\_type**
- typedef void **value\_type**
- typedef void **value\_type**

### Public Member Functions

- constexpr **allocator** (const allocator &\_\_a) noexcept
- constexpr **allocator** (const allocator< \_Tp1 > &) noexcept
- template<typename \_Up>  
constexpr **allocator** (const allocator< \_Up > &) noexcept
- constexpr void \* **allocate** (size\_t \_\_n)
- constexpr void \* **allocate** (size\_type \_\_n, const void \* \_\_p) noexcept
- constexpr void \* **allocate** (size\_type \_\_n, const void \* \_\_p) noexcept
- constexpr allocation\_result< void \*, size\_t > **allocate\_at\_least** (size\_t \_\_n)
- constexpr void **deallocate** (\_Tp \* \_\_p, size\_type \_\_n)
- constexpr void **deallocate** (void \* \_\_p, size\_t \_\_n)
- allocator & **operator=** (const allocator &) = default

### Static Public Member Functions

- static constexpr void **\_S\_check\_allocation\_limit** (size\_t \_\_n)
- static constexpr void **\_S\_check\_allocation\_limit** (size\_t \_\_n)

### Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

### Related Symbols

(Note that these are not member symbols.)

- constexpr bool **operator==** (const allocator< \_T1 > &, const allocator< \_T2 > &) noexcept
- constexpr bool **operator==** (const allocator< \_T1 > &, const allocator< \_T2 > &) noexcept

#### 6.188.1 Detailed Description

std::allocator<void> specialization.

#### 6.188.2 Friends And Related Symbol Documentation

##### **operator==( )** [1/2]

```
bool operator== (
 const allocator< _T1 > & ,
 const allocator< _T2 > &) [related]
```

Equality comparison for std::allocator objects

##### Returns

true, for all std::allocator objects.

**operator==()** [2/2]

```
bool operator== (
 const allocator<_T1> & ,
 const allocator<_T2> &) [related]
```

Equality comparison for std::allocator objects

**Returns**

true, for all std::allocator objects.

The documentation for this class was generated from the following file:

- [allocator.h](#)

**6.189 std::allocator\_traits<\_Alloc> Struct Template Reference**

```
#include <memory>
```

Inheritance diagram for `std::allocator_traits<_Alloc>`:



## Public Types

- typedef `_Alloc` `allocator_type`
- using `const_pointer`

- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- template<typename \_Tp>  
using **rebind\_alloc**
- template<typename \_Tp>  
using **rebind\_traits**
- using [size\\_type](#)
- typedef \_Alloc::value\_type [value\\_type](#)
- using [void\\_pointer](#)

### Static Public Member Functions

- static constexpr [pointer](#) [allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer](#) [allocate](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- static constexpr [allocation\\_result](#)< [pointer](#), [size\\_type](#) > [allocate\\_at\\_least](#) (\_Alloc &\_\_a, [size\\_type](#) \_\_n)
- template<typename \_Tp, typename... \_Args>  
static constexpr void [construct](#) (\_Alloc &\_\_a, \_Tp \*\_\_p, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_construct<\_Tp, \_Args... >())
- static constexpr void [deallocate](#) (\_Alloc &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- template<typename \_Tp>  
static constexpr void [destroy](#) (\_Alloc &\_\_a, \_Tp \*\_\_p) noexcept(\_S\_nothrow\_destroy<\_Tp >())
- static constexpr [size\\_type](#) [max\\_size](#) (const \_Alloc &\_\_a) noexcept
- static constexpr \_Alloc [select\\_on\\_container\\_copy\\_construction](#) (const \_Alloc &\_\_rhs)

#### 6.189.1 Detailed Description

template<typename \_Alloc>  
struct std::allocator\_traits<\_Alloc>

Uniform interface to all allocator types.

Since

C++11

#### 6.189.2 Member Typedef Documentation

##### allocator\_type

```
template<typename _Alloc>
typedef _Alloc std::allocator_traits<_Alloc>::allocator_type
```

The allocator type.

##### const\_pointer

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::const_pointer
```

The allocator's const pointer type.

[Alloc::const\\_pointer](#) if that type exists, otherwise [pointer\\_traits<pointer>::rebind<const value\\_type>](#)

**const\_void\_pointer**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

**difference\_type**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type
```

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

**is\_always\_equal**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

**pointer**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer
```

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

**propagate\_on\_container\_copy\_assignment**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_move\_assignment**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

**propagate\_on\_container\_swap**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

**size\_type**

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type
```

The allocator's size type.



Alloc::size\_type if that type exists, otherwise make\_unsigned<difference\_type>::type

### value\_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits<_Alloc>::value_type
The allocated type.
```

### void\_pointer

```
template<typename _Alloc>
using std::allocator_traits<_Alloc>::void_pointer
The allocator's void pointer type.
Alloc::void_pointer if that type exists, otherwise pointer_traits<pointer>::rebind<void>
```

## 6.189.3 Member Function Documentation

### allocate() [1/2]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits<_Alloc>::allocate (
 _Alloc & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

#### Parameters

|                          |                                              |
|--------------------------|----------------------------------------------|
| $\leftrightarrow$<br>__a | An allocator.                                |
| $\leftrightarrow$<br>__n | The number of objects to allocate space for. |

Calls a.allocate(n)

### allocate() [2/2]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits<_Alloc>::allocate (
 _Alloc & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

#### Parameters

|        |                                              |
|--------|----------------------------------------------|
| __a    | An allocator.                                |
| __n    | The number of objects to allocate space for. |
| __hint | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for  $n$  objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

**allocate\_at\_least()**

```
template<typename _Alloc>
constexpr allocation_result< pointer, size_type > std::allocator_traits< _Alloc >::allocate_at_least(
 _Alloc & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory, generously.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__a</code> | An allocator.                                        |
| <code>__n</code> | The minimum number of objects to allocate space for. |

**Returns**

Memory of suitable size and alignment for  $n$  or more contiguous objects of type `value_type`.

Returns `a.allocate_at_least(n)` if that expression is well-formed, else `{ a.allocate(n), n }`. When an allocator is obliged to reserve more space than required for the cited  $n$  objects, it may deliver the extra space to the caller.

**construct()**

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
constexpr void std::allocator_traits< _Alloc >::construct (
 _Alloc & __a,
 _Tp * __p,
 _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`.

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

**deallocate()**

```
template<typename _Alloc>
constexpr void std::allocator_traits< _Alloc >::deallocate (
```

```

 _Alloc & __a,
 pointer __p,
 size_type __n) [inline], [static], [constexpr]

```

Deallocate memory.

#### Parameters

|                     |                                                |
|---------------------|------------------------------------------------|
| $\leftarrow$<br>__a | An allocator.                                  |
| $\leftarrow$<br>__p | Pointer to the memory to deallocate.           |
| $\leftarrow$<br>__n | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

#### destroy()

```

template<typename _Alloc>
template<typename _Tp>
constexpr void std::allocator_traits<_Alloc>::destroy (
 _Alloc & __a,
 _Tp * __p) [inline], [static], [constexpr], [noexcept]

```

Destroy an object of type `_Tp`.

#### Parameters

|                     |                                  |
|---------------------|----------------------------------|
| $\leftarrow$<br>__a | An allocator.                    |
| $\leftarrow$<br>__p | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

#### max\_size()

```

template<typename _Alloc>
constexpr size_type std::allocator_traits<_Alloc>::max_size (
 const _Alloc & __a) [inline], [static], [constexpr], [noexcept]

```

The maximum supported allocation size.

#### Parameters

|                     |               |
|---------------------|---------------|
| $\leftarrow$<br>__a | An allocator. |
|---------------------|---------------|

#### Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

**select\_on\_container\_copy\_construction()**

```
template<typename _Alloc>
constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
 const _Alloc & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

**6.190 std::allocator\_traits< allocator< \_Tp > > Struct Template Reference**

```
#include <memory>
```

**Public Types**

- using [allocator\\_type](#)
- using [const\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- template<typename \_Up>
 using **rebind\_alloc**
- template<typename \_Up>
 using **rebind\_traits**
- using [size\\_type](#)
- using [value\\_type](#)
- using [void\\_pointer](#)

**Static Public Member Functions**

- static constexpr [pointer](#) [allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- static constexpr [pointer](#) [allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#) \_\_hint)
- static constexpr [allocation\\_result](#)< [pointer](#), [size\\_type](#) > [allocate\\_at\\_least](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- template<typename \_Up, typename... \_Args>
 static constexpr void [construct](#) ([allocator\\_type](#) &\_\_a, \_Up \*\_\_p, \_Args &&... \_\_args) noexcept(\_\_is\_nothrow\_↵
 new\_constructible< \_Up, \_Args... >)
- static constexpr void [deallocate](#) ([allocator\\_type](#) &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)

- `template<typename _Up>`  
`static constexpr void destroy (allocator\_type &__a, _Up *__p) noexcept(is\_nothrow\_destructible< _Up >::value)`
- `static constexpr size\_type max\_size (const allocator\_type &__a) noexcept`
- `static constexpr allocator\_type select\_on\_container\_copy\_construction (const allocator\_type &__rhs)`

### 6.190.1 Detailed Description

`template<typename _Tp>`  
`struct std::allocator\_traits< allocator< _Tp > >`

Partial specialization for `std::allocator`.

Since

C++11

See also

`std::allocator_traits`

### 6.190.2 Member Typedef Documentation

#### `allocator_type`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::allocator\_type`  
 The allocator type.

#### `const_pointer`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::const\_pointer`  
 The allocator's const pointer type.

#### `const_void_pointer`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::const\_void\_pointer`  
 The allocator's const void pointer type.

#### `difference_type`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::difference\_type`  
 The allocator's difference type.

#### `is_always_equal`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::is\_always\_equal`  
 Whether all instances of the allocator type compare equal.

#### `pointer`

`template<typename _Tp>`  
`using std::allocator\_traits< allocator< _Tp > >::pointer`  
 The allocator's pointer type.

**propagate\_on\_container\_copy\_assignment**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

**propagate\_on\_container\_move\_assignment**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

**propagate\_on\_container\_swap**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

**size\_type**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::size_type
```

The allocator's size type.

**value\_type**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::value_type
```

The allocated type.

**void\_pointer**

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

**6.190.3 Member Function Documentation****allocate() [1/2]**

```
template<typename _Tp>
constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
 allocator_type & __a,
 size_type __n) [inline], [static], [constexpr]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| $\leftarrow$<br>__a | An allocator.                                |
| $\leftarrow$<br>__n | The number of objects to allocate space for. |

Calls `a.allocate(n)`

**allocate()** [2/2]

```
template<typename _Tp>
constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
 allocator_type & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static], [constexpr]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

**allocate\_at\_least()**

```
template<typename _Tp>
constexpr allocation_result< pointer, size_type > std::allocator_traits< allocator< _Tp > >::←
allocate_at_least (
 allocator_type & __a,
 size_type __n) [inline], [static], [nodiscard], [constexpr]
```

Allocate memory, generously.

**Parameters**

|                        |                                                      |
|------------------------|------------------------------------------------------|
| <code>←<br/>__a</code> | An allocator.                                        |
| <code>←<br/>__n</code> | The minimum number of objects to allocate space for. |

**Returns**

Memory of suitable size and alignment for  $m \geq n$  contiguous objects of type `value_type`, and *m*.

Returns `a.allocate_at_least(n)`.

**construct()**

```
template<typename _Tp>
template<typename _Up, typename... _Args>
constexpr void std::allocator_traits< allocator< _Tp > >::construct (
 allocator_type & __a,
 _Up * __p,
 _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`.

#### Parameters

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args)...) in C++11, C++14 and C++17.`  
 Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...) instead.`  
 References [std::\\_Construct\(\)](#), and [std::forward\(\)](#).

#### deallocate()

```
template<typename _Tp>
constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
 allocator_type & __a,
 pointer __p,
 size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

#### Parameters

|                          |                                                |
|--------------------------|------------------------------------------------|
| <code>__↔<br/>__a</code> | An allocator.                                  |
| <code>__↔<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>__↔<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

#### destroy()

```
template<typename _Tp>
template<typename _Up>
constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
 allocator_type & __a,
 _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`.

#### Parameters

|                          |                                  |
|--------------------------|----------------------------------|
| <code>__↔<br/>__a</code> | An allocator.                    |
| <code>__↔<br/>__p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p).`



**max\_size()**

```
template<typename _Tp>
constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
 const allocator_type & __a) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

**Returns**

```
__a.max_size()
```

**select\_on\_container\_copy\_construction()**

```
template<typename _Tp>
constexpr allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container_copy_

construction (
 const allocator_type & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**

```
__rhs
```

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

**6.191 `std::allocator_traits< allocator< void > >` Struct Reference**

```
#include <memory>
```

**Public Types**

- using `allocator_type`
- using `const_pointer`
- using `const_void_pointer`
- using `difference_type`
- using `is_always_equal`
- using `pointer`
- using `propagate_on_container_copy_assignment`
- using `propagate_on_container_move_assignment`
- using `propagate_on_container_swap`
- template<typename \_Up>  
using `rebind_alloc`

- `template<typename _Up>`  
  using **rebind\_traits**
- using `size_type`
- using `value_type`
- using `void_pointer`

### Static Public Member Functions

- static void \* `allocate` (`allocator_type` &, `size_type`, const void \*`=nullptr`)=delete
- static `allocation_result< void *, size_type >` `allocate_at_least` (`allocator_type` &, `size_type`)=delete
- `template<typename _Up, typename... _Args>`  
  static constexpr void `construct` (`allocator_type` &, `_Up *``__p`, `_Args &&... __args`) noexcept(`__is_nothrow_newable< _constructible< _Up, _Args... >>`)
- static void `deallocate` (`allocator_type` &, void \*, `size_type`)=delete
- `template<typename _Up>`  
  static constexpr void `destroy` (`allocator_type` &, `_Up *``__p`) noexcept(`is_nothrow_destructible< _Up >::value`)
- static `size_type` `max_size` (const `allocator_type` &)=delete
- static constexpr `allocator_type` `select_on_container_copy_construction` (const `allocator_type` & `__rhs`)

#### 6.191.1 Detailed Description

Explicit specialization for `std::allocator<void>`.

Since

C++11

See also

`std::allocator_traits`

#### 6.191.2 Member Typedef Documentation

##### **allocator\_type**

using `std::allocator_traits< allocator< void > >::allocator_type`

The allocator type.

##### **const\_pointer**

using `std::allocator_traits< allocator< void > >::const_pointer`

The allocator's const pointer type.

##### **const\_void\_pointer**

using `std::allocator_traits< allocator< void > >::const_void_pointer`

The allocator's const void pointer type.

##### **difference\_type**

using `std::allocator_traits< allocator< void > >::difference_type`

The allocator's difference type.

##### **is\_always\_equal**

using `std::allocator_traits< allocator< void > >::is_always_equal`

Whether all instances of the allocator type compare equal.

**pointer**

using `std::allocator_traits< allocator< void > >::pointer`  
 The allocator's pointer type.

**propagate\_on\_container\_copy\_assignment**

using `std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment`  
 How the allocator is propagated on copy assignment.

**propagate\_on\_container\_move\_assignment**

using `std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment`  
 How the allocator is propagated on move assignment.

**propagate\_on\_container\_swap**

using `std::allocator_traits< allocator< void > >::propagate_on_container_swap`  
 How the allocator is propagated on swap.

**size\_type**

using `std::allocator_traits< allocator< void > >::size_type`  
 The allocator's size type.

**value\_type**

using `std::allocator_traits< allocator< void > >::value_type`  
 The allocated type.

**void\_pointer**

using `std::allocator_traits< allocator< void > >::void_pointer`  
 The allocator's void pointer type.

**6.191.3 Member Function Documentation****allocate()**

```
void * std::allocator_traits< allocator< void > >::allocate (
 allocator_type & ,
 size_type ,
 const void * = nullptr) [static], [delete]
```

allocate is ill-formed for allocator<void>

**construct()**

```
template<typename _Up, typename... _Args>
constexpr void std::allocator_traits< allocator< void > >::construct (
 allocator_type & ,
 _Up * __p,
 _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`.

**Parameters**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args)...) in C++11, C++14 and C++17. Changed in C++20 to call std::construct_at(__p, std::forward<_Args>(__args)...) instead. References std::\_Construct\(\), and std::forward\(\).`

**deallocate()**

```
void std::allocator_traits< allocator< void > >::deallocate (
 allocator_type & ,
 void * ,
 size_type) [static], [delete]
```

`deallocate` is ill-formed for `allocator<void>`

**destroy()**

```
template<typename _Up>
constexpr void std::allocator_traits< allocator< void > >::destroy (
 allocator_type & ,
 _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__↵<br/>_a</code> | An allocator.                    |
| <code>__↵<br/>_p</code> | Pointer to the object to destroy |

Invokes the destructor for `*__p`.

References [std::\\_Destroy\(\)](#).

**max\_size()**

```
size_type std::allocator_traits< allocator< void > >::max_size (
 const allocator_type &) [static], [delete]
```

`max_size` is ill-formed for `allocator<void>`

**select\_on\_container\_copy\_construction()**

```
constexpr allocator_type std::allocator_traits< allocator< void > >::select_on_container_copy_↵
construction (
 const allocator_type & __rhs) [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**`__rhs`

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

**6.192 `std::allocator_traits< pmr::polymorphic_allocator< _Tp > >` Struct Template Reference**

```
#include <memory_resource>
```

**Public Types**

- using [allocator\\_type](#)
- using [const\\_pointer](#)
- using [const\\_void\\_pointer](#)
- using [difference\\_type](#)
- using [is\\_always\\_equal](#)
- using [pointer](#)
- template<typename \_Up>  
using **rebind\_alloc**
- template<typename \_Up>  
using **rebind\_traits**
- using [size\\_type](#)
- using [value\\_type](#)
- using [void\\_pointer](#)

**Static Public Member Functions**

- static [pointer allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- static [pointer allocate](#) ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n, [const\\_void\\_pointer](#))
- static `std::allocation_result< pointer, size\_type > allocate_at_least` ([allocator\\_type](#) &\_\_a, [size\\_type](#) \_\_n)
- template<typename \_Up, typename... \_Args>  
static void [construct](#) ([allocator\\_type](#) &\_\_a, \_Up \*\_\_p, \_Args &&... \_\_args)
- static void [deallocate](#) ([allocator\\_type](#) &\_\_a, [pointer](#) \_\_p, [size\\_type](#) \_\_n)
- template<typename \_Up>  
static constexpr void [destroy](#) ([allocator\\_type](#) &, \_Up \*\_\_p) noexcept([is\\_nothrow\\_destructible](#)< \_Up >::value)
- static constexpr [size\\_type max\\_size](#) (const [allocator\\_type](#) &) noexcept
- using [propagate\\_on\\_container\\_copy\\_assignment](#)
- using [propagate\\_on\\_container\\_move\\_assignment](#)
- using [propagate\\_on\\_container\\_swap](#)
- static [allocator\\_type select\\_on\\_container\\_copy\\_construction](#) (const [allocator\\_type](#) &) noexcept

### 6.192.1 Detailed Description

```
template<typename _Tp>
struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >
```

Partial specialization for `std::pmr::polymorphic_allocator`.

Since

C++17

### 6.192.2 Member Typedef Documentation

#### **allocator\_type**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocator_type
The allocator type.
```

#### **const\_pointer**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
The allocator's const pointer type.
```

#### **const\_void\_pointer**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
The allocator's const void pointer type.
```

#### **difference\_type**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
The allocator's difference type.
```

#### **is\_always\_equal**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
Whether all instances of the allocator type compare equal.
```

#### **pointer**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
The allocator's pointer type.
```

#### **propagate\_on\_container\_copy\_assignment**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↵
assignment
A polymorphic_allocator does not propagate when a container is copied, moved, or swapped.
```

**propagate\_on\_container\_move\_assignment**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↵
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**propagate\_on\_container\_swap**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

**size\_type**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.

**value\_type**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::value_type
```

The allocated type.

**void\_pointer**

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

**6.192.3 Member Function Documentation****allocate() [1/2]**

```
template<typename _Tp>
pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
 allocator_type & __a,
 size_type __n) [inline], [static], [nodiscard]
```

Allocate memory.

**Parameters**

|                        |                                              |
|------------------------|----------------------------------------------|
| <code>↵<br/>__a</code> | An allocator.                                |
| <code>↵<br/>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`.

**allocate() [2/2]**

```
template<typename _Tp>
pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
 allocator_type & __a,
```

```

 size_type __n,
 const_void_pointer) [inline], [static], [nodiscard]

```

Allocate memory.

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>↵<br/>_a</code> | An allocator.                                |
| <code>↵<br/>_n</code> | The number of objects to allocate space for. |

#### Returns

Memory of suitable size and alignment for `n` objects of type `value_type`.

The third parameter is ignored..

Returns `a.allocate(n)`.

#### `allocate_at_least()`

```

template<typename _Tp>
std::allocation_result< pointer, size_type > std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::allocate_at_least (
 allocator_type & __a,
 size_type __n) [inline], [static], [nodiscard]

```

Allocate memory.

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>↵<br/>_a</code> | An allocator.                                |
| <code>↵<br/>_n</code> | The number of objects to allocate space for. |

#### Returns

Memory of suitable size and alignment for `n` objects of type `value_type`.

Just returns `{ a.allocate(n), n }::polymorphic_allocator` cannot be extended without breaking ABI.

#### `construct()`

```

template<typename _Tp>
template<typename _Up, typename... _Args>
void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
 allocator_type & __a,
 _Up * __p,
 _Args &&... __args) [inline], [static]

```

Construct an object of type `_Up`.



**Parameters**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args) ...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args) ...)` instead. References [std::forward\(\)](#).

**deallocate()**

```
template<typename _Tp>
void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
 allocator_type & __a,
 pointer __p,
 size_type __n) [inline], [static]
```

Deallocate memory.

**Parameters**

|                        |                                                |
|------------------------|------------------------------------------------|
| <code>↔<br/>__a</code> | An allocator.                                  |
| <code>↔<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>↔<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`.

**destroy()**

```
template<typename _Tp>
template<typename _Up>
constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
 allocator_type & ,
 _Up * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`.

**Parameters**

|                        |                                  |
|------------------------|----------------------------------|
| <code>↔<br/>__a</code> | An allocator.                    |
| <code>↔<br/>__p</code> | Pointer to the object to destroy |

Calls `p->_Up()`.

**max\_size()**

```
template<typename _Tp>
constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size (
 const allocator_type &) [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

**Returns**

```
numeric_limits<size_t>::max() / sizeof(value_type)
```

**select\_on\_container\_copy\_construction()**

```
template<typename _Tp>
allocator_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::select_on_container↵
_copy_construction (
 const allocator_type &) [inline], [static], [noexcept]
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

The documentation for this struct was generated from the following file:

- [memory\\_resource.h](#)

**6.193 \_\_gnu\_cxx::limit\_condition::always\_adjustor Struct Reference**

```
#include <throw_allocator.h>
```

**6.193.1 Detailed Description**

Always enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**6.194 \_\_gnu\_cxx::random\_condition::always\_adjustor Struct Reference**

```
#include <throw_allocator.h>
```

**6.194.1 Detailed Description**

Always enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**6.195 \_\_gnu\_cxx::annotate\_base Struct Reference**

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::annotate_base`:



### Public Member Functions

- void **check** (size\_t label)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)

### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

### Friends

- [std::ostream](#) & **operator<<** ([std::ostream](#) &os, const annotate\_base &\_\_b)

#### 6.195.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size. The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.196 std::experimental::fundamentals\_v1::any Class Reference

```
#include <any>
```

## Public Member Functions

- [any](#) () noexcept
- template<typename \_ValueType, typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename [enable\\_if](#)<![is\\_constructible](#)<\_Tp, \_ValueType && >::value, bool >::type = false>  
[any](#) (\_ValueType &&\_\_value)
- template<typename \_ValueType, typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename [enable\\_if](#)<[is\\_constructible](#)<\_Tp, \_ValueType && >::value, bool >::type = true>  
[any](#) (\_ValueType &&\_\_value)
- [any](#) (any &&\_\_other) noexcept
- [any](#) (const any &\_\_other)
- [~any](#) ()
- void [clear](#) () noexcept
- bool [empty](#) () const noexcept
- template<typename \_ValueType>  
[enable\\_if](#) <![is\\_same](#)< [any](#), [decay\\_t](#)< \_ValueType > >::value, [any](#) & > [operator=](#) (\_ValueType &&\_\_rhs)
- [any](#) & [operator=](#) (any &&\_\_rhs) noexcept
- [any](#) & [operator=](#) (const [any](#) &\_\_rhs)
- void [swap](#) (any &\_\_rhs) noexcept
- const [type\\_info](#) & [type](#) () const noexcept

## Static Public Member Functions

- template<typename \_Tp>  
static constexpr bool [\\_\\_is\\_valid\\_cast](#) ()

## Friends

- template<typename \_Tp>  
[enable\\_if](#) < [is\\_object](#)< \_Tp >::value, void \* > [\\_\\_any\\_caster](#) (const [any](#) \*\_\_any)

### 6.196.1 Detailed Description

A type-safe container of any type.

An [any](#) object's state is either empty or it stores a contained object of CopyConstructible type.

### 6.196.2 Constructor & Destructor Documentation

#### [any](#)() [1/5]

```
std::experimental::fundamentals_v1::any::any () [inline], [noexcept]
```

Default constructor, creates an empty object.

Referenced by [any\(\)](#), [any\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [swap\(\)](#).

#### [any](#)() [2/5]

```
std::experimental::fundamentals_v1::any::any (
 const any & __other) [inline]
```

Copy constructor, copies the state of \_\_other.

References [any\(\)](#), and [empty\(\)](#).

**any()** [3/5]

```
std::experimental::fundamentals_v1::any::any (
 any && __other) [inline], [noexcept]
```

Move constructor, transfer the state from `__other`.

**Postcondition**

`__other.empty()` (this postcondition is a GNU extension)

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

**any()** [4/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
std::experimental::fundamentals_v1::any::any (
 _ValueType && __value) [inline]
```

Construct with a copy of `__value` as the contained object.

References [std::forward\(\)](#).

**any()** [5/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
std::experimental::fundamentals_v1::any::any (
 _ValueType && __value) [inline]
```

Construct with a copy of `__value` as the contained object.

**~any()**

```
std::experimental::fundamentals_v1::any::~~any () [inline]
```

Destructor, calls `clear()`.

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

**6.196.3 Member Function Documentation****clear()**

```
void std::experimental::fundamentals_v1::any::clear () [inline], [noexcept]
```

If not empty, destroy the contained object.

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

Referenced by [~any\(\)](#), and [operator=\(\)](#).

**empty()**

```
bool std::experimental::fundamentals_v1::any::empty () const [inline], [nodiscard], [noexcept]
```

Reports whether there is a contained object or not.

Referenced by [any\(\)](#), [clear\(\)](#), [swap\(\)](#), and [type\(\)](#).

**operator=()** [1/3]

```
template<typename _ValueType>
enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > std::experimental::fundamentals_v1::any::operator= (
 _ValueType && __rhs) [inline]
```

Store a copy of `__rhs` as the contained object.

References [any\(\)](#), and [std::forward\(\)](#).

**operator=()** [2/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
 any && __rhs) [inline], [noexcept]
```

Move assignment operator.

Postcondition

`__rhs.empty()` (not guaranteed for other implementations)

References [any\(\)](#), and [clear\(\)](#).

**operator=()** [3/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
 const any & __rhs) [inline]
```

Copy the state of another object.

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

**swap()**

```
void std::experimental::fundamentals_v1::any::swap (
 any & __rhs) [inline], [noexcept]
```

Exchange state with another object.

References [any\(\)](#), and [empty\(\)](#).

**type()**

```
const type_info & std::experimental::fundamentals_v1::any::type () const [inline], [noexcept]
```

The typeid of the contained object, or typeid(void) if empty.

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

The documentation for this class was generated from the following file:

- [experimental/any](#)

**6.197 std::array< \_Tp, \_Nm > Struct Template Reference**

```
#include <array>
```

**Public Types**

- typedef const value\_type \* **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

**Public Member Functions**

- constexpr reference **at** (size\_type \_\_n)
- constexpr const\_reference **at** (size\_type \_\_n) const
- constexpr const\_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr pointer **data** () noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr void **fill** (const value\_type &\_\_u)
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- constexpr reference **operator[]** (size\_type \_\_n) noexcept
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr reverse\_iterator **rbegin** () noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept
- constexpr reverse\_iterator **rend** () noexcept
- constexpr size\_type **size** () const noexcept
- constexpr void **swap** ([array](#) &\_\_other) noexcept(\_\_array\_traits<\_Tp, \_Nm >::\_Is\_nothrow\_swappable::value)

**Public Attributes**

- \_\_array\_traits<\_Tp, \_Nm >::\_Type **\_M\_elems**

**6.197.1 Detailed Description**

**template<typename \_Tp, std::size\_t \_Nm>**

**struct std::array<\_Tp, \_Nm >**

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

**Template Parameters**

|           |                                                  |
|-----------|--------------------------------------------------|
| <i>Tp</i> | Type of element. Required to be a complete type. |
| <i>Nm</i> | Number of elements.                              |

The documentation for this struct was generated from the following file:

- [array](#)

## 6.198 \_\_gnu\_pbds::associative\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::associative\_tag:



### 6.198.1 Detailed Description

Basic associative-container.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.199 std::atomic<\_Tp> Struct Template Reference

```
#include <atomic>
```

### Public Types

- using **value\_type**

### Public Member Functions

- constexpr **atomic** (*\_Tp* \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_strong** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (*\_Tp* &\_\_e, *\_Tp* \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- *\_Tp* **exchange** (*\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- *\_Tp* **exchange** (*\_Tp* \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- *\_Tp* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- *\_Tp* **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_Tp** () const noexcept
- **operator \_Tp** () const volatile noexcept
- *\_Tp* **operator=** (*\_Tp* \_\_i) noexcept



- `_Tp operator= (_Tp __i) volatile noexcept`
- `atomic & operator= (const atomic &) volatile =delete`
- `atomic & operator= (const atomic &)=delete`
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

### 6.199.1 Detailed Description

**template<typename \_Tp>**  
**struct std::atomic<\_Tp>**

Generic atomic type, primary class template.

#### Template Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>_Tp</code> | Type to be made atomic, must be trivially copyable. |
|------------------|-----------------------------------------------------|

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.200 std::atomic<\_Tp \* > Struct Template Reference

```
#include <atomic>
```

### Public Types

- `typedef __atomic_base<_Tp * > __base_type`
- `typedef _Tp * __pointer_type`
- using `difference_type`
- using `value_type`

### Public Member Functions

- `constexpr atomic (__pointer_type __p) noexcept`
- `atomic (const atomic &)=delete`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`

- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_max (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_max (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_min (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_min (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __pointer_type () const noexcept`
- `operator __pointer_type () const volatile noexcept`
- `operator _Tp () const noexcept`
- `__pointer_type operator++ () noexcept`
- `__pointer_type operator++ () volatile noexcept`
- `__pointer_type operator++ (int) noexcept`
- `__pointer_type operator++ (int) volatile noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator-- () noexcept`
- `__pointer_type operator-- () volatile noexcept`
- `__pointer_type operator-- (int) noexcept`
- `__pointer_type operator-- (int) volatile noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator= (__pointer_type __p) noexcept`
- `__pointer_type operator= (__pointer_type __p) volatile noexcept`
- `atomic & operator= (const atomic &) volatile =delete`
- `atomic & operator= (const atomic &) =delete`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`

## Public Attributes

- `__base_type _M_b`

### 6.200.1 Detailed Description

```
template<typename _Tp>
struct std::atomic< _Tp * >
```

Partial specialization for pointer types.

The documentation for this struct was generated from the following file:

- `atomic`

## 6.201 std::atomic< bool > Struct Reference

```
#include <atomic>
```

## Public Types

- using **value\_type**

## Public Member Functions

- constexpr **atomic** (bool \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator bool** () const noexcept
- **operator bool** () const noexcept
- **operator bool** () const volatile noexcept
- **operator bool** () const volatile noexcept
- bool **operator=** (bool \_\_i) noexcept
- bool **operator=** (bool \_\_i) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

### 6.201.1 Detailed Description

atomic<bool>

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.202 std::atomic< char > Struct Reference

```
#include <atomic>
```

## Public Types

- typedef \_\_atomic\_base< char > **\_\_base\_type**
- typedef char **\_\_integral\_type**
- using **value\_type**

## Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (char &\_\_e, char \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (char &\_\_e, char \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- char **exchange** (char \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- char **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator char** () const noexcept
- **operator char** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.202.1 Detailed Description

Explicit specialization for char.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.203 std::atomic< char16\_t > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< char16\_t > **\_\_base\_type**
- typedef char16\_t **\_\_integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (char16\_t &\_\_e, char16\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (char16\_t &\_\_e, char16\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- char16\_t **exchange** (char16\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- char16\_t **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator char16\_t** () const noexcept
- **operator char16\_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char16\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.203.1 Detailed Description

Explicit specialization for char16\_t.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.204 std::atomic< char32\_t > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< char32\_t > **\_\_base\_type**
- typedef char32\_t **\_\_integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (char32\_t &\_\_e, char32\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (char32\_t &\_\_e, char32\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- char32\_t **exchange** (char32\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- char32\_t **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator char32\_t** () const noexcept
- **operator char32\_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char32\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

#### 6.204.1 Detailed Description

Explicit specialization for char32\_t.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.205 std::atomic< int > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< int > **\_\_base\_type**
- typedef int **\_\_integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (int &\_\_e, int \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (int &\_\_e, int \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- int **exchange** (int \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- int **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator int** () const noexcept
- **operator int** () const volatile noexcept

- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (int \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

#### 6.205.1 Detailed Description

Explicit specialization for int.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 6.206 std::atomic< long > Struct Reference

```
#include <atomic>
```

#### Public Types

- typedef \_\_atomic\_base< long > **\_\_base\_type**
- typedef long **\_\_integral\_type**
- using **value\_type**

#### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (long &\_\_e, long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (long &\_\_e, long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- long **exchange** (long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- long **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator long** () const noexcept
- **operator long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

#### 6.206.1 Detailed Description

Explicit specialization for long.

The documentation for this struct was generated from the following file:

- [atomic](#)

### 6.207 std::atomic< long long > Struct Reference

```
#include <atomic>
```

#### Public Types

- typedef \_\_atomic\_base< long long > **\_\_base\_type**
- typedef long long **\_\_integral\_type**
- using **value\_type**

**Public Member Functions**

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (long long &\_\_e, long long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (long long &\_\_e, long long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- long long **exchange** (long long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- long long **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator long long** () const noexcept
- **operator long long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (long long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

**6.207.1 Detailed Description**

Explicit specialization for long long.

The documentation for this struct was generated from the following file:

- [atomic](#)

**6.208 std::atomic< short > Struct Reference**

```
#include <atomic>
```

**Public Types**

- typedef \_\_atomic\_base< short > **\_\_base\_type**
- typedef short **\_\_integral\_type**
- using **value\_type**

**Public Member Functions**

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (short &\_\_e, short \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (short &\_\_e, short \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- short **exchange** (short \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- short **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator short** () const noexcept
- **operator short** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (short \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

**6.208.1 Detailed Description**

Explicit specialization for short.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.209 `std::atomic< signed char >` Struct Reference

```
#include <atomic>
```

### Public Types

- typedef `__atomic_base< signed char >` `__base_type`
- typedef `signed char` `__integral_type`
- using `value_type`

### Public Member Functions

- constexpr **atomic** (`__integral_type __i`) noexcept
- **atomic** (`const atomic &`)=delete
- bool **compare\_exchange\_strong** (`signed char &__e`, `signed char __i`, [memory\\_order \\_\\_s](#), [memory\\_order \\_\\_f](#)) noexcept
- bool **compare\_exchange\_weak** (`signed char &__e`, `signed char __i`, [memory\\_order \\_\\_s](#), [memory\\_order \\_\\_f](#)) noexcept
- `signed char` **exchange** (`signed char __i`, [memory\\_order \\_\\_m=memory\\_order\\_seq\\_cst](#)) noexcept
- bool **is\_lock\_free** () const noexcept
- `signed char` **load** ([memory\\_order \\_\\_m=memory\\_order\\_seq\\_cst](#)) const noexcept
- **operator signed char** () const noexcept
- **operator signed char** () const volatile noexcept
- `atomic &` **operator=** (`const atomic &`) volatile =delete
- `atomic &` **operator=** (`const atomic &`)=delete
- void **store** (`signed char __i`, [memory\\_order \\_\\_m=memory\\_order\\_seq\\_cst](#)) noexcept

### 6.209.1 Detailed Description

Explicit specialization for signed char.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.210 `std::atomic< unsigned char >` Struct Reference

```
#include <atomic>
```

### Public Types

- typedef `__atomic_base< unsigned char >` `__base_type`
- typedef `unsigned char` `__integral_type`
- using `value_type`

### Public Member Functions

- constexpr **atomic** (`__integral_type __i`) noexcept
- **atomic** (`const atomic &`)=delete
- bool **compare\_exchange\_strong** (`unsigned char &__e`, `unsigned char __i`, [memory\\_order \\_\\_s](#), [memory\\_order \\_\\_f](#)) noexcept
- bool **compare\_exchange\_weak** (`unsigned char &__e`, `unsigned char __i`, [memory\\_order \\_\\_s](#), [memory\\_order \\_\\_f](#)) noexcept
- `unsigned char` **exchange** (`unsigned char __i`, [memory\\_order \\_\\_m=memory\\_order\\_seq\\_cst](#)) noexcept
- bool **is\_lock\_free** () const noexcept



- unsigned char **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator unsigned char** () const noexcept
- **operator unsigned char** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned char \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.210.1 Detailed Description

Explicit specialization for unsigned char.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.211 `std::atomic< unsigned int >` Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< unsigned int > **\_\_base\_type**
- typedef unsigned int **\_\_integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (unsigned int &\_\_e, unsigned int \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (unsigned int &\_\_e, unsigned int \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- unsigned int **exchange** (unsigned int \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- unsigned int **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator unsigned int** () const noexcept
- **operator unsigned int** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned int \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.211.1 Detailed Description

Explicit specialization for unsigned int.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.212 `std::atomic< unsigned long >` Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< unsigned long > \_\_**base\_type**
- typedef unsigned long \_\_**integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (unsigned long &\_\_e, unsigned long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (unsigned long &\_\_e, unsigned long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- unsigned long **exchange** (unsigned long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- unsigned long **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator unsigned long** () const noexcept
- **operator unsigned long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

#### 6.212.1 Detailed Description

Explicit specialization for unsigned long.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 6.213 std::atomic< unsigned long long > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< unsigned long long > \_\_**base\_type**
- typedef unsigned long long \_\_**integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (unsigned long long &\_\_e, unsigned long long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (unsigned long long &\_\_e, unsigned long long \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- unsigned long long **exchange** (unsigned long long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- unsigned long long **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator unsigned long long** () const noexcept
- **operator unsigned long long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned long long \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.213.1 Detailed Description

Explicit specialization for unsigned long long.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.214 std::atomic< unsigned short > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< unsigned short > **\_\_base\_type**
- typedef unsigned short **\_\_integral\_type**
- using **value\_type**

### Public Member Functions

- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (unsigned short &\_\_e, unsigned short \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (unsigned short &\_\_e, unsigned short \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- unsigned short **exchange** (unsigned short \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- unsigned short **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator unsigned short** () const noexcept
- **operator unsigned short** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned short \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.214.1 Detailed Description

Explicit specialization for unsigned short.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.215 std::atomic< wchar\_t > Struct Reference

```
#include <atomic>
```

### Public Types

- typedef \_\_atomic\_base< wchar\_t > **\_\_base\_type**
- typedef wchar\_t **\_\_integral\_type**
- using **value\_type**

## Public Member Functions

- constexpr **atomic** ([\\_\\_integral\\_type](#) \_\_i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare\_exchange\_strong** (wchar\_t &\_\_e, wchar\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (wchar\_t &\_\_e, wchar\_t \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- wchar\_t **exchange** (wchar\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **is\_lock\_free** () const noexcept
- wchar\_t **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- **operator wchar\_t** () const noexcept
- **operator wchar\_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (wchar\_t \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept

### 6.215.1 Detailed Description

Explicit specialization for wchar\_t.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 6.216 std::atomic\_flag Struct Reference

```
#include <atomic_base.h>
```

## Public Member Functions

- constexpr **atomic\_flag** (bool \_\_i) noexcept
- **atomic\_flag** (const atomic\_flag &)=delete
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **clear** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- atomic\_flag & **operator=** (const atomic\_flag &) volatile =delete
- atomic\_flag & **operator=** (const atomic\_flag &)=delete
- bool **test** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- bool **test** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **test\_and\_set** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

### 6.216.1 Detailed Description

atomic\_flag

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 6.217 std::auto\_ptr< \_Tp > Class Template Reference

```
#include <memory>
```

## Public Types

- typedef \_Tp [element\\_type](#)

**Public Member Functions**

- [auto\\_ptr](#) (auto\_ptr &\_\_a) throw ()
- template<typename \_Tp1>  
  [auto\\_ptr](#) (auto\_ptr<\_Tp1> &\_\_a) throw ()
- [auto\\_ptr](#) (auto\_ptr\_ref< [element\\_type](#) > \_\_ref) throw ()
- [auto\\_ptr](#) ([element\\_type](#) \*\_\_p=0) throw ()
- ~[auto\\_ptr](#) ()
- [element\\_type](#) \* [get](#) () const throw ()
- template<typename \_Tp1>  
  **operator** [auto\\_ptr](#)<\_Tp1> () throw ()
- template<typename \_Tp1>  
  **operator** [auto\\_ptr\\_ref](#)<\_Tp1> () throw ()
- [element\\_type](#) & [operator\\*](#) () const throw ()
- [element\\_type](#) \* [operator->](#) () const throw ()
- [auto\\_ptr](#) & [operator=](#) ([auto\\_ptr](#) &\_\_a) throw ()
- template<typename \_Tp1>  
  [auto\\_ptr](#) & [operator=](#) ([auto\\_ptr](#)<\_Tp1> &\_\_a) throw ()
- [auto\\_ptr](#) & [operator=](#) (auto\_ptr\_ref< [element\\_type](#) > \_\_ref) throw ()
- [element\\_type](#) \* [release](#) () throw ()
- void [reset](#) ([element\\_type](#) \*\_\_p=0) throw ()

**6.217.1 Detailed Description**

**template<typename \_Tp>**  
**class std::auto\_ptr<\_Tp>**

A simple smart pointer providing strict ownership semantics.  
 The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.  
[\\_GLIBCXX\\_RESOLVE\\_LIB\\_DEFECTS](#)

1. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

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

**6.217.2 Member Typedef Documentation****element\_type**

```
template<typename _Tp>
typedef _Tp std::auto_ptr<_Tp>::element_type
```

The pointed-to type.


### 6.217.3 Constructor & Destructor Documentation

#### auto\_ptr() [1/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 element_type * __p = 0) throw () [inline], [explicit]
```

An auto\_ptr is usually constructed from a raw pointer.

#### Parameters

|                                                                                          |                               |
|------------------------------------------------------------------------------------------|-------------------------------|
| <br>__p | A pointer (defaults to NULL). |
|------------------------------------------------------------------------------------------|-------------------------------|


This object now *owns* the object pointed to by \_\_p.  
Referenced by [auto\\_ptr\(\)](#), [auto\\_ptr\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

#### auto\_ptr() [2/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr< _Tp > & __a) throw () [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

|                                                                                            |                                    |
|--------------------------------------------------------------------------------------------|------------------------------------|
| <br>__a | Another auto_ptr of the same type. |
|--------------------------------------------------------------------------------------------|------------------------------------|


This object now *owns* the object previously owned by \_\_a, which has given up ownership.  
References [auto\\_ptr\(\)](#).

#### auto\_ptr() [3/4]

```
template<typename _Tp>
template<typename _Tp1>
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr< _Tp1 > & __a) throw () [inline]
```

An auto\_ptr can be constructed from another auto\_ptr.

#### Parameters

|                                                                                            |                                                   |
|--------------------------------------------------------------------------------------------|---------------------------------------------------|
| <br>__a | Another auto_ptr of a different but related type. |
|--------------------------------------------------------------------------------------------|---------------------------------------------------|

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.  
This object now *owns* the object previously owned by \_\_a, which has given up ownership.  
References [auto\\_ptr\(\)](#).

#### ~auto\_ptr()

```
template<typename _Tp>
std::auto_ptr< _Tp >::~~auto_ptr () [inline]
```

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., get () is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if \_Tp::~~Tp() throws, but this is prohibited. [17.4.3.6]/2

### auto\_ptr() [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr_ref< element_type > __ref) throw () [inline]
```

Automatic conversions.

These operations are supposed to convert an auto\_ptr into and from an auto\_ptr\_ref automatically as needed. This would allow constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(...);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(...);
```

But it doesn't work, and won't be fixed. For further details see <https://cplusplus.github.io/LWG/↵lwg-closed.html#463>

## 6.217.4 Member Function Documentation

### get()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::get () const throw () [inline]
```

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This auto\_ptr still owns the memory.

### operator\*()

```
template<typename _Tp>
element_type & std::auto_ptr< _Tp >::operator* () const throw () [inline]
```

Smart pointer dereferencing.

If this auto\_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

### operator->()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::operator-> () const throw () [inline]
```

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

### operator=() [1/2]

```
template<typename _Tp>
auto_ptr & std::auto_ptr< _Tp >::operator= (
 auto_ptr< _Tp > & __a) throw () [inline]
```

`auto_ptr` assignment operator.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of the same type. |
|------------------|-------------------------------------------------|

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

References [auto\\_ptr\(\)](#), and [reset\(\)](#).

#### `operator=()` [2/2]

```
template<typename _Tp>
template<typename _Tp1>
auto_ptr & std::auto_ptr< _Tp >::operator= (
 auto_ptr< _Tp1 > & __a) throw () [inline]
```

`auto_ptr` assignment operator.

#### Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__a</code> | Another <code>auto_ptr</code> of a different but related type. |
|------------------|----------------------------------------------------------------|

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp/element_type`.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

References [auto\\_ptr\(\)](#), and [reset\(\)](#).

#### `release()`

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::release () throw () [inline]
```

Bypassing the smart pointer.

#### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

#### Note

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

#### `reset()`

```
template<typename _Tp>
void std::auto_ptr< _Tp >::reset (
 element_type * __p = 0) throw () [inline]
```

Forcibly deletes the managed object.



**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__p</code> | A pointer (defaults to NULL). |
|------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Referenced by [operator=\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following files:

- [shared\\_ptr\\_base.h](#)
- [auto\\_ptr.h](#)

**6.218 std::auto\_ptr\_ref< \_Tp1 > Struct Template Reference**

```
#include <auto_ptr.h>
```

**Public Member Functions**

- `auto_ptr_ref(_Tp1 * __p)`

**Public Attributes**

- `_Tp1 * _M_ptr`

**6.218.1 Detailed Description**

```
template<typename _Tp1>
struct std::auto_ptr_ref< _Tp1 >
```

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

**6.219 std::back\_insert\_iterator< \_Container > Class Template Reference**

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::back_insert_iterator<_Container>`:



### Public Types

- typedef `_Container` `container_type`
- using `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- constexpr `back_insert_iterator` (`_Container` &\_\_x)
- constexpr `back_insert_iterator` & `operator*` ()
- constexpr `back_insert_iterator` & `operator++` ()
- constexpr `back_insert_iterator` `operator++` (int)
- constexpr `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- constexpr `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container` \* `container`

#### 6.219.1 Detailed Description

**template<typename \_Container>**  
**class** `std::back_insert_iterator<_Container>`

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

### 6.219.2 Member Typedef Documentation

#### container\_type

```
template<typename _Container>
```

```
typedef _Container std::back_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

#### iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
```

```
iterator_category [inherited]
```

One of the [tag types](#).

#### pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

#### reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

#### value\_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

### 6.219.3 Constructor & Destructor Documentation

#### back\_insert\_iterator()

```
template<typename _Container>
```

```
std::back_insert_iterator< _Container >::back_insert_iterator (
 _Container & __x) [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.

References [std::\\_\\_addressof\(\)](#).

### 6.219.4 Member Function Documentation

#### operator\*()

```
template<typename _Container>
```

```
back_insert_iterator & std::back_insert_iterator< _Container >::operator* () [inline], [nodiscard],
[constexpr]
```

Simply returns \*this.

#### operator++() [1/2]

```
template<typename _Container>
```

```
back_insert_iterator & std::back_insert_iterator< _Container >::operator++ () [inline], [constexpr]
```

Simply returns \*this. (This iterator does not *move*.)

**operator++()** [2/2]

```
template<typename _Container>
back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
 int) [inline], [constexpr]
```

Simply returns `*this`. (This iterator does not *move*).

**operator=()**

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator= (
 const typename _Container::value_type & __value) [inline], [constexpr]
```

**Parameters**

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**6.220 std::bad\_alloc Class Reference**

```
#include <new_except.h>
```

Inheritance diagram for `std::bad_alloc`:

**Public Member Functions**

- `_GLIBCXX26_CONSTEXPR bad_alloc (const bad_alloc &)=default`
- `_GLIBCXX26_CONSTEXPR bad_alloc & operator= (const bad_alloc &)=default`
- `virtual constexpr const char * what () const noexcept`

### 6.220.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

### 6.220.2 Member Function Documentation

#### `what()`

```
virtual constexpr const char * std::bad_alloc::what () const [inline], [constexpr], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [new\\_except.h](#)

## 6.221 std::experimental::fundamentals\_v1::bad\_any\_cast Class Reference

```
#include <any>
```

Inheritance diagram for `std::experimental::fundamentals_v1::bad_any_cast`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

### 6.221.1 Detailed Description

Exception class thrown by a failed `any_cast`.

### 6.221.2 Member Function Documentation

#### what()

```
virtual const char * std::experimental::fundamentals_v1::bad_any_cast::what () const [inline],
[virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::bad\\_cast](#).

The documentation for this class was generated from the following file:

- [experimental/any](#)

### 6.222 std::bad\_cast Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad\_cast:



#### Public Member Functions

- virtual constexpr const char \* [what](#) () const noexcept

#### 6.222.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

#### 6.222.2 Member Function Documentation

#### what()

```
virtual constexpr const char * std::bad_cast::what () const [inline], [constexpr], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#).

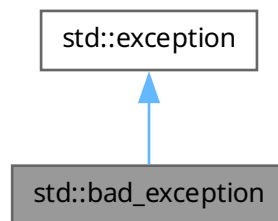
The documentation for this class was generated from the following file:

- [typeinfo](#)

## 6.223 `std::bad_exception` Class Reference

```
#include <exception>
```

Inheritance diagram for `std::bad_exception`:



### Public Member Functions

- virtual constexpr const char \* [what](#) () const noexcept

#### 6.223.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

#### 6.223.2 Member Function Documentation

##### `what()`

```
virtual constexpr const char * std::bad_exception::what () const [inline], [constexpr], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

## 6.224 `std::bad_function_call` Class Reference

```
#include <std_function.h>
```

Inheritance diagram for `std::bad_function_call`:



### Public Member Functions

- `const char * what () const noexcept`

#### 6.224.1 Detailed Description

Exception class thrown when class template function's `operator()` is called with an empty target.

#### 6.224.2 Member Function Documentation

##### `what()`

```
const char * std::bad_function_call::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

### 6.225 `std::experimental::fundamentals_v1::bad_optional_access` Class Reference

```
#include <optional>
```



Inheritance diagram for `std::experimental::fundamentals_v1::bad_optional_access`:



### Public Member Functions

- **`bad_optional_access`** (`const char *__arg`)
- virtual `const char * what () const` `noexcept`

#### 6.225.1 Detailed Description

Exception class thrown when a disengaged optional object is dereferenced.

#### 6.225.2 Member Function Documentation

##### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

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

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [experimental/optional](#)

## 6.226 `std::bad_typeid` Class Reference

```
#include <typeinfo>
```

Inheritance diagram for `std::bad_typeid`:



### Public Member Functions

- virtual constexpr const char \* [what](#) () const noexcept

#### 6.226.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

#### 6.226.2 Member Function Documentation

##### `what()`

```
virtual constexpr const char * std::bad_typeid::what () const [inline], [constexpr], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

#### 6.227 `std::bad_weak_ptr` Class Reference

```
#include <shared_ptr_base.h>
```

Inheritance diagram for `std::bad_weak_ptr`:



### Public Member Functions

- virtual `char const * what () const` `noexcept`

#### 6.227.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

#### 6.227.2 Member Function Documentation

##### `what()`

```
virtual char const * std::bad_weak_ptr::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

### 6.228 \_\_gnu\_parallel::balanced\_quicksort\_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



### Public Member Functions

- `balanced_quicksort_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

#### 6.228.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

#### 6.228.2 Member Function Documentation

##### `__get_num_threads()`

`_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline], [inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

void `__gnu_parallel::parallel_tag::set_num_threads` (  
     `_ThreadIndex` `__num_threads`) [inline], [inherited]

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.229 `__gnu_parallel::balanced_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::balanced_tag`:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 6.229.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

#### 6.229.2 Member Function Documentation

##### `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads() [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

##### `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads(
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.230 `std::tr2::bases<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- typedef `__reflection_typelist<__bases(_Tp)...>` **type**

#### 6.230.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::bases<_Tp>
```

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 6.231 `__gnu_pbds::basic_branch<Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc>` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_branch<Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc>`:



### Public Types

- typedef `Node_Update` **node\_update**

### Protected Member Functions

- **basic\_branch** (const basic\_branch &other)
- template<typename T0>  
**basic\_branch** (T0 t0)
- template<typename T0, typename T1>  
**basic\_branch** (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>  
**basic\_branch** (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)

- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

### 6.231.1 Detailed Description

`template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>`

**class \_\_gnu\_pbds::basic\_branch**< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >

A branched, tree-like (tree, trie) container abstraction.

#### Template Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                           |
| <i>Mapped</i>      | Map type.                                                           |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> . |
| <i>Node_Update</i> | Updates nodes, restores invariants.                                 |
| <i>Policy_Tl</i>   | Policy typelist.                                                    |
| <i>_Alloc</i>      | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 6.232 \_\_gnu\_pbds::basic\_branch\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_branch_tag`:



### 6.232.1 Detailed Description

Basic branch structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.233 std::basic\_filebuf<\_CharT, \_Traits> Class Template Reference

```
#include <fstream>
```

Inheritance diagram for `std::basic_filebuf<_CharT, _Traits>`:



### Public Types

- typedef `codecvt< char_type, char, __state_type > __codecvt_type`
- typedef `__basic_file< char > __file_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `traits_type::state_type __state_type`
- typedef `basic_streambuf< char_type, traits_type > __streambuf_type`
- typedef `_CharT char_type`
- typedef `traits_type::int_type int_type`
- typedef `traits_type::off_type off_type`
- typedef `traits_type::pos_type pos_type`
- typedef `_Traits traits_type`

### Public Member Functions

- `basic_filebuf()`
- `basic_filebuf(basic_filebuf &&)`
- `basic_filebuf(const basic_filebuf &)=delete`
- virtual `~basic_filebuf()`
- `__filebuf_type * close()`
- `locale getloc() const`
- `streamsize in_avail()`
- `bool is_open() const throw()`
- `template<typename _Path>`  
`_If_fs_path<_Path, __filebuf_type * > open(const _Path &__s, ios_base::openmode __mode)`
- `__filebuf_type * open(const char *__s, ios_base::openmode __mode)`
- `__filebuf_type * open(const std::string &__s, ios_base::openmode __mode)`



- `basic_filebuf` & `operator=` (`basic_filebuf` &&)
- `basic_filebuf` & `operator=` (const `basic_filebuf` &)=delete
- `locale pubimbue` (const `locale` &\_\_loc)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- `streamsize sgetn` (char\_type \*\_\_s, `streamsize` \_\_n)
- `int_type snextc` ()
- `int_type sputbackc` (char\_type \_\_c)
- `int_type sputc` (char\_type \_\_c)
- `streamsize sputn` (const char\_type \*\_\_s, `streamsize` \_\_n)
- `int_type sungetc` ()
- void `swap` (`basic_filebuf` &)
- `basic_streambuf` \* `pubsetbuf` (char\_type \*\_\_s, `streamsize` \_\_n)
- `pos_type pubseekoff` (off\_type \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- `pos_type pubseekpos` (pos\_type \_\_sp, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- `int pubsync` ()

### Protected Member Functions

- void `__safe_gbump` (`streamsize` \_\_n)
- void `__safe_pbump` (`streamsize` \_\_n)
- void `_M_allocate_internal_buffer` ()
- bool `_M_convert_to_external` (char\_type \*, `streamsize`)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- int `_M_get_ext_pos` (\_\_state\_type &\_\_state)
- `pos_type _M_seek` (off\_type \_\_off, `ios_base::seekdir` \_\_way, \_\_state\_type \_\_state)
- void `_M_set_buffer` (`streamsize` \_\_off)
- bool `_M_terminate_output` ()
- void `gbump` (int \_\_n)
- virtual void `imbue` (const `locale` &\_\_loc)
- virtual `int_type overflow` (int\_type \_\_c=\_Traits::eof())
- virtual `int_type pbackfail` (int\_type \_\_c=\_Traits::eof())
- void `pbump` (int \_\_n)
- virtual `pos_type seekoff` (off\_type \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- virtual `pos_type seekpos` (pos\_type \_\_pos, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
- virtual `__streambuf_type * setbuf` (char\_type \*\_\_s, `streamsize` \_\_n)
- void `setg` (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
- void `setp` (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
- virtual `streamsize showmanyc` ()
- void `swap` (`basic_streambuf` &\_\_sb)
- virtual `int sync` ()
- virtual `int_type uflow` ()
- virtual `int_type underflow` ()
- virtual `streamsize xsgetn` (char\_type \*\_\_s, `streamsize` \_\_n)
- virtual `streamsize xspn` (const char\_type \*\_\_s, `streamsize` \_\_n)
- char\_type \* `eback` () const

- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const

- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

### Protected Attributes

- char\_type \* [\\_M\\_buf](#)
  - bool [\\_M\\_buf\\_allocated](#)
  - [locale](#) [\\_M\\_buf\\_locale](#)
  - size\_t [\\_M\\_buf\\_size](#)
  - const \_\_codecvt\_type \* [\\_M\\_codecvt](#)
  - char \* [\\_M\\_ext\\_buf](#)
  - [streamsize](#) [\\_M\\_ext\\_buf\\_size](#)
  - char \* [\\_M\\_ext\\_end](#)
  - const char \* [\\_M\\_ext\\_next](#)
  - \_\_file\_type [\\_M\\_file](#)
  - char\_type \* [\\_M\\_in\\_beg](#)
  - char\_type \* [\\_M\\_in\\_cur](#)
  - char\_type \* [\\_M\\_in\\_end](#)
  - \_\_c\_lock [\\_M\\_lock](#)
  - [ios\\_base::openmode](#) [\\_M\\_mode](#)
  - char\_type \* [\\_M\\_out\\_beg](#)
  - char\_type \* [\\_M\\_out\\_cur](#)
  - char\_type \* [\\_M\\_out\\_end](#)
  - bool [\\_M\\_reading](#)
  - \_\_state\_type [\\_M\\_state\\_beg](#)
  - \_\_state\_type [\\_M\\_state\\_cur](#)
  - \_\_state\_type [\\_M\\_state\\_last](#)
  - bool [\\_M\\_writing](#)
- 
- char\_type [\\_M\\_pback](#)
  - char\_type \* [\\_M\\_pback\\_cur\\_save](#)
  - char\_type \* [\\_M\\_pback\\_end\\_save](#)
  - bool [\\_M\\_pback\\_init](#)

### Friends

- class [ios\\_base](#)

### 6.233.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >
```

The actual work of input and output (for files).

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on `traits_type`, specific to this class:

- `traits_type::pos_type` must be `fpos<traits_type::state_type>`
- `traits_type::off_type` must be `streamoff`
- `traits_type::state_type` must be Assignable and DefaultConstructible,
- `traits_type::state_type()` must be the initial state for `codecvt`.

### 6.233.2 Constructor & Destructor Documentation

#### `basic_filebuf()`

```
template<typename _CharT, typename _Traits>
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ()
```

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

References `_M_buf`, `std::basic_streambuf< _CharT, _Traits >::_M_buf_locale`, `_M_buf_size`, `_M_ext_buf`, `_M_ext_buf_size`, `_M_ext_next`, `_M_mode`, `_M_pback`, `_M_pback_cur_save`, `_M_pback_end_save`, `_M_pback_init`, and `_M_reading`.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `close()`.

#### `~basic_filebuf()`

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf () [inline], [virtual]
```

The destructor closes the file first.

### 6.233.3 Member Function Documentation

#### `_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback () [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Referenced by `pbackfail()`.

#### `_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw () [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Referenced by `overflow()`, `seekoff()`, `seekpos()`, `underflow()`, and `xsgntrn()`.

**\_M\_set\_buffer()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
 streamsize __off) [inline], [protected]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

\_\_off == egptr() - eback() upon underflow/uflow (**read** mode); \_\_off == 0 upon overflow (**write** mode); \_\_off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == \_M\_buf\_size - 1, since \_M\_buf\_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Referenced by [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

**close()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()
```

Closes the currently associated file.

**Returns**

this on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

References [basic\\_filebuf\(\)](#), [\\_M\\_mode](#), [\\_M\\_pback\\_init](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), and [is\\_open\(\)](#).

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::~~basic\\_filebuf\(\)](#), and [open\(\)](#).

**eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_destroy\\_pback\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputbackc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**egptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence

- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_create\\_pback\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::in\\_avail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sgetc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_stringbuf< char\\_type, traits\\_type >::sputc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

### epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

### gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

### Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputbackc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::uflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

### getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
 const locale & __loc) [protected], [virtual]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_ext_buf`, `_M_ext_next`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `is_open()`, and `seekoff()`.

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::is_open () const throw () [inline], [nodiscard]
```

Returns true if the external file is open.

Referenced by [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [close\(\)](#), [imbue\(\)](#), [open\(\)](#), [seekoff\(\)](#), [seekpos\(\)](#), [setbuf\(\)](#), and [showmanyc\(\)](#).

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

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path<_Path, __filebuf_type * > std::basic_filebuf<_CharT, _Traits>::open (
 const _Path & __s,
 ios_base::openmode __mode) [inline]
```

Opens an external file.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The name of the file, as a filesystem::path. |
| <code>__mode</code> | The open mode flags.                         |

**Returns**

`this` on success, NULL on failure

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

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open (
 const char * __s,
 ios_base::openmode __mode)
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

|         |          |                  |                  |  |
|---------|----------|------------------|------------------|--|
| +-----+ |          |                  |                  |  |
|         | ios_base | Flag combination | stdio equivalent |  |
|         | binary   | in out trunc app |                  |  |





**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_buf\\_size](#), [\\_M\\_destroy\\_pback\(\)](#), [\\_M\\_mode](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::app](#), [std::ios\\_base::cur](#), [std::ios\\_base::out](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

Referenced by [pbackfail\(\)](#), [sync\(\)](#), [underflow\(\)](#), and [xsgetn\(\)](#).

**pbackfail()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_create\\_pback\(\)](#), [\\_M\\_mode](#), [\\_M\\_pback\\_init](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::cur](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::eback\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gbump\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::in](#), [std::ios\\_base::in](#), [overflow\(\)](#), [seekoff\(\)](#), and [underflow\(\)](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits >::str\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::sync\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::str\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived `imbue(__loc)`.

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekoff` function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#).

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns [uflow\(\)](#).

Referenced by [std::basic\\_iostream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::snextc\(\)](#).

**seekoff()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_destroy\\_pback\(\)](#), [\\_M\\_reading](#), [std::ios\\_base::cur](#), [is\\_open\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

Referenced by [imbue\(\)](#), [open\(\)](#), and [pbackfail\(\)](#).

**seekpos()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_destroy\\_pback\(\)](#), [std::ios\\_base::beg](#), and [is\\_open\(\)](#).

**setbuf()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >::↵
setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                           |
|------------------------|---------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area. |
| <code>↵<br/>__n</code> | Size of __s.              |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html↵  
#io.streambuf.buffering](https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering) for more.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

References [\\_M\\_buf](#), [\\_M\\_buf\\_size](#), and [is\\_open\(\)](#).

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_create\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_destroy\\_pback\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_set\\_buffer\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
```

```
char_type * __pbeg,
char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

#### Postcondition

```
__pbeg == pbase(), __pbeg == pptr(), and __pend == epptr()
```

Referenced by `std::wbuffer_convert<_Codecv, _Elem, _Tr>::wbuffer_convert()`, and `std::basic_filebuf<char_type, traits_type>::_M_s`

#### sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_iostream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ign` and `std::basic_streambuf<char_type, traits_type>::snextc()`.

#### sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsggetn`.

#### Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsggetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

#### showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::showmanyc () [protected], [virtual]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::ios_base::in`, and `is_open()`.

**snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, char_traits< _CharT > >::get()`, and `std::basic_istream< _CharT, char_traits< _CharT > >::readsome()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

### Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

### sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

### Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xsgputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

### sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

### sync()

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync () [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.



**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [overflow\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), and [xsgetn\(\)](#).

**underflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or [traits::eof\(\)](#) to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns [eof\(\)](#).

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_buf\\_size](#), [\\_M\\_destroy\\_pback\(\)](#), [\\_M\\_mode](#), [\\_M\\_set\\_buffer\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::eback\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gptr\(\)](#), [std::ios\\_base::in](#), and [overflow\(\)](#).

Referenced by [pbackfail\(\)](#).

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character extraction.

**Parameters**

|                                                                                                                    |                                         |
|--------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| <a href="#"> <code>__s</code></a> | A buffer area.                          |
| <a href="#"> <code>__n</code></a> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_buf\\_size](#), [\\_M\\_destroy\\_pback\(\)](#), [\\_M\\_mode](#), [\\_M\\_pback\\_init](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gbump\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::in](#), [std::ios\\_base::in](#), [overflow\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::setg\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::xsgetn\(\)](#).

**xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xspn (
 const char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character insertion.

**Parameters**

|                                                                                                                      |                                        |
|----------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"> <code>__s</code></a> | A buffer area.                         |
| <a href="#"> <code>__n</code></a> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_buf\\_size](#), [\\_M\\_mode](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::app](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::eptr\(\)](#), [std::ios\\_base::out](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::xspn\(\)](#).

#### 6.233.4 Member Data Documentation

##### \_M\_buf

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf [protected]
```

Pointer to the beginning of internal buffer.

Referenced by [basic\\_filebuf\(\)](#), and [setbuf\(\)](#).

##### \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::basic\\_filebuf\(\)](#).

##### \_M\_buf\_size

```
template<typename _CharT, typename _Traits>
```

```
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [setbuf\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

##### \_M\_ext\_buf

```
template<typename _CharT, typename _Traits>
```

```
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

##### \_M\_ext\_buf\_size

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected]
```

Size of buffer held by `_M_ext_buf`.

Referenced by [basic\\_filebuf\(\)](#).

##### \_M\_ext\_next

```
template<typename _CharT, typename _Traits>
```

```
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

##### \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

**\_M\_in\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
Current read area.
```

**\_M\_in\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
End of get area.
```

**\_M\_mode**

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current filebuf.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [showmanyc\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

**\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
Start of put area.
```

**\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
Current put area.
```

**\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
End of put area.
```

**\_M\_pback**

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**\_M\_pback\_cur\_save**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**`_M_pback_end_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf<_CharT, _Traits>::_M_pback_end_save [protected]
Necessary bits for putback buffer management.
```

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

**`_M_pback_init`**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::_M_pback_init [protected]
Necessary bits for putback buffer management.
```

**Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#), [close\(\)](#), [pbackfail\(\)](#), and [xsgetn\(\)](#).

**`_M_reading`**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::_M_reading [protected]
_M_reading == false && _M_writing == false for uncommitted mode; _M_reading == true for read mode; _M_writing
== true for write mode;
```

NB: `_M_reading == true && _M_writing == true` is unused.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [seekoff\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)

**6.234 `std::basic_fstream<_CharT, _Traits>` Class Template Reference**

```
#include <fstream>
```

Inheritance diagram for `std::basic_fstream<_CharT, _Traits>`:



## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_filebuf<char_type, traits_type>` `__filebuf_type`
- typedef `basic_ios<char_type, traits_type>` `__ios_type`
- typedef `basic_iostream<char_type, traits_type>` `__iostream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event __e, ios_base &__b, int __i)`
- typedef `_ios_Fmtflags` `fmtflags`

- typedef traits\_type::int\_type **int\_type**
- typedef \_ios\_istate **istate**
- typedef traits\_type::off\_type **off\_type**
- typedef \_ios\_Openmode **openmode**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_ios\_Seekdir **seekdir**
- typedef \_Traits **traits\_type**

## Public Member Functions

- **basic\_fstream** ()
- **basic\_fstream** (basic\_fstream &&\_\_rhs)
- template<typename \_Path, typename \_Require = \_If\_fs\_path<\_Path>>  
  **basic\_fstream** (const \_Path &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **basic\_fstream** (const basic\_fstream &)=delete
- **basic\_fstream** (const char \* \_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **basic\_fstream** (const std::string &\_\_s, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- ~**basic\_fstream** ()
- template<typename \_ValueT>  
  **basic\_istream**<\_CharT, \_Traits> & **\_M\_extract** (\_ValueT &\_\_v)
- const locale & **\_M\_getloc** () const
- template<typename \_ValueT>  
  **basic\_ostream**<\_CharT, \_Traits> & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_setstate** (istate \_\_state)
- bool **bad** () const
- void **clear** (istate \_\_state=goodbit)
- void **close** ()
- **basic\_ios** & **copyfmt** (const **basic\_ios** &\_\_rhs)
- bool **eof** () const
- istate **exceptions** () const
- void **exceptions** (istate \_\_except)
- bool **fail** () const
- char\_type **fill** () const
- char\_type **fill** (char\_type \_\_ch)
- fmtflags **flags** () const
- fmtflags **flags** (fmtflags \_\_fmtfl)
- \_\_ostream\_type & **flush** ()
- streamsize **gcount** () const
- **basic\_istream**<wchar\_t> & **getline** (char\_type \* \_\_s, streamsize \_\_n, char\_type \_\_delim)
- **basic\_istream**<char> & **getline** (char\_type \* \_\_s, streamsize \_\_n, char\_type \_\_delim)
- locale **getloc** () const
- bool **good** () const
- **basic\_istream**<wchar\_t> & **ignore** (streamsize \_\_n)
- **basic\_istream**<char> & **ignore** (streamsize \_\_n)
- **basic\_istream**<wchar\_t> & **ignore** (streamsize \_\_n, int\_type \_\_delim)
- **basic\_istream**<char> & **ignore** (streamsize \_\_n, int\_type \_\_delim)
- locale **imbue** (const locale &\_\_loc)
- bool **is\_open** ()
- bool **is\_open** () const
- long & **iword** (int \_\_ix)
- char **narrow** (char\_type \_\_c, char \_\_dfault) const

- `template<typename _Path>`  
`_If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `__ostream_type & operator<< ( __streambuf_type * __sb)`
  - `__ostream_type & operator<< (const void * __p)`
  - `__ostream_type & operator<< (const volatile void * __p)`
  - `__ostream_type & operator<< (nullptr_t)`
  - `basic_fstream & operator= (basic_fstream && __rhs)`
  - `basic_fstream & operator= (const basic_fstream &)=delete`
  - `__istream_type & operator>> ( __streambuf_type * __sb)`
  - `__istream_type & operator>> (void *& __p)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `__filebuf_type * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `__ostream_type & seekp (pos_type)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void swap (basic_fstream & __rhs)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> ( __istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> ( __ios_type &(*__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (int & __n)`
- `__istream_type & operator>> (unsigned int & __n)`



- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n, char __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator bool () const`
- `bool operator! () const`

- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

*All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

*If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.*

*If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.*

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type * __s, streamsize __n)`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`

- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- void `_M_cache_locale` (const [locale](#) &\_\_loc)
- void `_M_call_callbacks` ([event](#) \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename \_ValueT>  
\_\_istream\_type & `_M_extract` (\_ValueT &\_\_v)
- \_Words & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT>  
\_\_ostream\_type & `_M_insert` (\_ValueT \_\_v)
- void `_M_move` ([ios\\_base](#) &) noexcept
- void `_M_swap` ([ios\\_base](#) &\_\_rhs) noexcept
- void `init` ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void `move` ([basic\\_ios](#) &&\_\_rhs)
- void `move` ([basic\\_ios](#) &\_\_rhs)
- void `set_rdbuf` ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void `swap` ([basic\\_ios](#) &\_\_rhs) noexcept
- void `swap` ([basic\\_istream](#) &\_\_rhs)
- void `swap` ([basic\\_istream](#) &\_\_rhs)
- void `swap` ([basic\\_ostream](#) &\_\_rhs)

### Protected Attributes

- \_Callback\_list \* `_M_callbacks`
- const \_\_ctype\_type \* `_M_ctype`
- [iostate](#) `_M_exception`
- char\_type `_M_fill`
- bool `_M_fill_init`
- [fmtflags](#) `_M_flags`
- [streamsize](#) `_M_gcount`
- [locale](#) `_M_ios_locale`
- \_Words `_M_local_word` [`_S_local_word_size`]
- const \_\_num\_get\_type \* `_M_num_get`
- const \_\_num\_put\_type \* `_M_num_put`
- [streamsize](#) `_M_precision`

- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### 6.234.1 Detailed Description

`template<typename _CharT, typename _Traits>`

`class std::basic_fstream<_CharT, _Traits>`

Controlling input and output for files.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

### 6.234.2 Member Typedef Documentation

#### `event_callback`

`typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

#### `fmtflags`

`typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`

- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.234.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**6.234.4 Constructor & Destructor Documentation****basic\_fstream() [1/4]**

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_fstream() [2/4]**

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

**basic\_fstream() [3/4]**

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.  |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

**basic\_fstream() [4/4]**

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Create an input/output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | filesystem::path specifying the filename.        |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

**~basic\_fstream()**

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::~~basic_fstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

**6.234.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#), and [std::time\\_put< \\_CharT, \\_Outlter >::put\(\)](#).

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::basic\\_fstream<char>::open\(\)](#), [std::basic\\_fstream<char>::open\(\)](#), [std::basic\\_ifstream<char>::open\(\)](#), [std::basic\\_ifstream<char>::open\(\)](#), [std::basic\\_ofstream<char>::open\(\)](#), [std::basic\\_ofstream<char>::open\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::seekg\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**close()**

```
template<typename _CharT, typename _Traits>
void std::basic_fstream<_CharT, _Traits>::close () [inline]
```

Close the file.

Calls [std::basic\\_filebuf::close\(\)](#). If that function fails, [failbit](#) is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that [rdbuf\(\)](#) and [rdstate\(\)](#) remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).



**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

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

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

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

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, and `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

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

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::chrono::operator<<()`, `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>`.

Referenced by `std::flush()`.

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_istream<char_type, traits_type>::get()`, and `std::basic_istream<char_type, traits_type>::get()`.

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >](#):

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s, __n, widen("\n"))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | Pointer to an array. |
|------------------|----------------------|

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

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

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s, __n, widen("\n"))`.

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

```
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, and `std::ios_base::goodbit`.

Referenced by `std::basic_istream< char_type, traits_type >::getline()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::operator<<()`, `std::operator<>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

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

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::ignore\(\)](#).

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

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

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

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 char __delim) [inline], [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

**ignore()** [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.





```
std::basic_istream< char >::basic_istream(), std::basic_istream< char >::basic_istream(),
std::basic_istream< char >::basic_istream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostream< char >::basic_ostream(), std::basic_ostream< char >::basic_ostream(),
std::basic_ostream< char >::basic_ostream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().
```

### is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_fstream< _CharT, _Traits >::is_open () [inline], [nodiscard]
```

Wrapper to test for an open file.

### Returns

```
rdbuf()->is_open()
```

### isword()

```
long & std::ios_base::isword (
 int __ix) [inline], [inherited]
```

Access to integer array.

### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

### Returns

A reference to an integer associated with the index.

The `isword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

Referenced by [isword\(\)](#).

### narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

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

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path<_Path, void> std::basic_fstream<_CharT, _Traits>::open (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The name of the file, as a filesystem::path. |
| <code>__mode</code> | The open mode flags.                         |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

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

```
template<typename _CharT, typename _Traits>
void std::basic_fstream<_CharT, _Traits>::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

Referenced by `std::basic_fstream<char>::basic_fstream()`, `std::basic_fstream<char>::basic_fstream()`, and `std::basic_fstream<char>::open()`.

**open() [3/3]**

```
template<typename _CharT, typename _Traits>
void std::basic_fstream<_CharT, _Traits>::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(* __pf) (__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets `failbit` in the error state

If the function inserts no characters, `failbit` is set.

### `operator<<()` [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

### `operator<<()` [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

### `operator<<()` [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

#### Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

#### Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

#### Parameters

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

#### Returns

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outiter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```



Integer arithmetic inserters.

#### Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

#### **operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(* __pf)(__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

#### **operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

#### **Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

#### **operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

#### **Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↩        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

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

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                                                                                                                       |                          |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a><br><code>__c</code> | The character to insert. |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)  
Referenced by [std::endl\(\)](#), and [std::ends\(\)](#).

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                                                                                                                         |                                                   |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| <a href="#"></a><br><code>__c</code> | The character to push back into the input stream. |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#),  
[std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdstate\(\)](#),  
[std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::sputbackc\(\)](#).

Referenced by [std::operator<<\(\)](#).



**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_fstream<_CharT, _Traits>::rdbuf () const [inline], [nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current basic\_filebuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument rdbuf(), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

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

**rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::`

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_istream< _CharT, _Traits >::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

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

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

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

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

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

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__mask</code> | The flags mask for <i>fmtfl</i> . |
|---------------------|-----------------------------------|

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

### setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

### sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, in).

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount(). At variance with putback, unget and seekg, eofbit is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, out).

References `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

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

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

**ungetc()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ungetc (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sungetc\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.



**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios<char_type, traits_type>::fill()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F` and `std::tr2::operator>>()`.

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

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width() [2/2]**

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.234.6 Member Data Documentation****`_M_gcount`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_iostream<char_type, traits_type>::operator<<()`, `std::basic_iostream<char_type, traits_type>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_iostream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_iostream< char_type, traits_type >::seekg()`, and `std::ws()`.

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >`  
and `std::basic_ostream< _CharT, _Traits >::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ws\(\)](#).

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_istreamstream< char >::basic_istreamstream()`, `std::basic_stringbuf< char >::basic_stringbuf()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_istreamstream< char >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekpos()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_streambuf< char_type, traits_type >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< char >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< char >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::left()`, and `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0`.

**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ostringstream< char >::basic\_ostringstream(), std::basic\_stringbuf< char >::basic\_stringbuf(), std::basic\_stringstream< char >::basic\_stringstream(), std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_streambuf< char\_type, traits\_type >::pubseekoff(), std::basic\_streambuf< char\_type, traits\_type >::pubseekpos(), std::basic\_streambuf< char\_type, traits\_type >::seekoff(), std::basic\_streambuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits >::seekp(), std::basic\_streambuf< char\_type, traits\_type >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits >::seekpos(), std::basic\_stringbuf< char >::seekpos(), std::basic\_ostringstream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputc.

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

## showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put<_CharT, _Outiter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

### showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

### showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

### skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x9E3779B9, 0x9E3779B9 >::operator>>\(\)](#), and [std::skipws\(\)](#).

### trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

### unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [fstream](#)

## 6.235 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash,`

Tag, Policy\_Tl, \_Alloc >:



### Protected Member Functions

- **basic\_hash\_table** (const basic\_hash\_table &other)
- template<typename T0>  
**basic\_hash\_table** (T0 t0)
- template<typename T0, typename T1>  
**basic\_hash\_table** (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

#### 6.235.1 Detailed Description

```

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy,
bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag,
Policy_Tl, _Alloc >

```

A hashed container abstraction.



**Template Parameters**

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                            |
| <i>Mapped</i>        | Map type.                                                            |
| <i>Hash_Fn</i>       | Hashing functor.                                                     |
| <i>Eq_Fn</i>         | Equal functor.                                                       |
| <i>Resize_Policy</i> | Resizes hash.                                                        |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. |
| <i>Tag</i>           | Instantiating data structure type, see <code>container_tag</code> .  |
| <i>Policy_TL</i>     | Policy typelist.                                                     |
| <i>_Alloc</i>        | Allocator type.                                                      |

Base is dispatched at compile time via `Tag`, from the following choices: `cc_hash_tag`, `gp_hash_tag`, and descendants of `basic_hash_tag`.

Base choices are: `detail::cc_ht_map`, `detail::gp_ht_map`

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

**6.236 `__gnu_pbds::basic_hash_tag` Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:

**6.236.1 Detailed Description**

Basic hash structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.237 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic\_ifstream< \_CharT, \_Traits >:



### Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [basic\\_istream](#)< char\_type, traits\_type > **\_\_istream\_type**
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { **erase\_event** , **imbue\_event** , **copyfmt\_event** }
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags **fmtflags**
- typedef traits\_type::int\_type **int\_type**
- typedef \_ios\_ostate **iostate**
- typedef traits\_type::off\_type **off\_type**
- typedef \_ios\_Openmode **openmode**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_ios\_Seekdir [seekdir](#)

- typedef `_Traits` **traits\_type**
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

## Public Member Functions

- [basic\\_ifstream](#) ()
- **basic\_ifstream** (basic\_ifstream &&\_rhs)
- template<typename \_Path, typename \_Require = \_If\_fs\_path<\_Path>>  
[basic\\_ifstream](#) (const \_Path &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_ifstream** (const basic\_ifstream &)=delete
- [basic\\_ifstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [~basic\\_ifstream](#) ()
- template<typename \_ValueT>  
[basic\\_istream](#)<\_CharT, \_Traits> & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags flags](#) () const
- [fmtflags flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize gcount](#) () const
- [basic\\_istream](#)< wchar\_t > & **getline** (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [basic\\_istream](#)< char > & **getline** (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale imbue](#) (const [locale](#) &\_\_loc)
- bool [is\\_open](#) ()
- bool **is\_open** () const
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- template<typename \_Path>  
\_If\_fs\_path<\_Path, void> **open** (const \_Path &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- void **open** (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- void **open** (const [std::string](#) &\_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) & **operator=** (basic\_ifstream &&\_rhs)
- [basic\\_ifstream](#) & **operator=** (const [basic\\_ifstream](#) &)=delete
- `__istream_type` & **operator>>** (`__streambuf_type` \* \_\_sb)

- `__istream_type & operator>> (void *&__p)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__filebuf_type * rdbuf () const`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `void swap (basic_ifstream & __rhs)`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- int\_type [get](#) ()
  - \_\_istream\_type & [get](#) (char\_type &\_\_c)
  - \_\_istream\_type & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - \_\_istream\_type & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - \_\_istream\_type & [get](#) (\_\_streambuf\_type &\_\_sb, char\_type \_\_delim)
  - \_\_istream\_type & [get](#) (\_\_streambuf\_type &\_\_sb)
  - \_\_istream\_type & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - \_\_istream\_type & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n)
  - \_\_istream\_type & [ignore](#) ()
  - \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n, char \_\_delim)
  - int\_type [peek](#) ()
  - \_\_istream\_type & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - \_\_istream\_type & [putback](#) (char\_type \_\_c)
  - \_\_istream\_type & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - \_\_istream\_type & [seekg](#) (pos\_type)
  - \_\_istream\_type & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [openmode](#) [\\_\\_noreplace](#)
- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)

- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename [\\_ValueT](#)>  
  [\\_\\_istream\\_type](#) & [\\_M\\_extract](#) ([\\_ValueT](#) &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void [move](#) ([basic\\_ios](#) &&\_\_rhs)
- void [move](#) ([basic\\_ios](#) &\_\_rhs)
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void [swap](#) ([basic\\_ios](#) &\_\_rhs) noexcept
- void [swap](#) ([basic\\_istream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 6.237.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ifstream< _CharT, _Traits >
```

Controlling input for files.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

### 6.237.2 Member Typedef Documentation

#### `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type [inherited]
```

These are non-standard types.

#### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

#### `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`

- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### **istate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc



**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.237.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**6.237.4 Constructor & Destructor Documentation****basic\_ifstream() [1/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ifstream<_CharT, _Traits>::basic_ifstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_ifstream() [2/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ifstream<_CharT, _Traits>::basic_ifstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

**basic\_ifstream() [3/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ifstream<_CharT, _Traits>::basic_ifstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | std::string specifying the filename.             |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::in` is automatically included in `__mode`.

**basic\_ifstream()** [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Create an input file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | filesystem::path specifying the filename.        |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::in` is automatically included in `__mode`.

**~basic\_ifstream()**

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

**6.237.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_put< _CharT, _Outlter >::do_put()`, and `std::time_put< _CharT, _Outlter >::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::seekg()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**close()**

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream<_CharT, _Traits>::close () [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

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

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

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

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#).

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

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

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

Referenced by [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), and [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>::`

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

`*this`

Returns `get(__s,__n,widen('\n'))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.



**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

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

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                                                                                                                       |                                               |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <a href="#"></a><br><code>__s</code> | A character array in which to store the data. |
| <a href="#"></a><br><code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

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

```
basic_ifstream< char > & std::basic_ifstream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, and `std::ios_base::goodbit`.

Referenced by `std::basic_istream< char_type, traits_type >::getline()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::do_get()`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::__detail::operator>>()`.

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

```
template<typename _CharT, typename _Traits>
```

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

Referenced by `std::basic_istream< char_type, traits_type >::ignore()`.

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

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

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

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream< _CharT, _Traits >::ignore (
 streamsize __n,
 char __delim) [inline], [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

**ignore()** [4/4]

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

`*this`

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)



**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>_↔<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

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

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path<_Path, void> std::basic_ifstream<_CharT, _Traits>::open (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__s</code> | The name of the file, as a <code>filesystem::path</code> . |
|------------------|------------------------------------------------------------|

|                     |                      |
|---------------------|----------------------|
| <code>__mode</code> | The open mode flags. |
|---------------------|----------------------|

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

### **open()** [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

#### **Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Referenced by [std::basic\\_ifstream< char >::basic\\_ifstream\(\)](#), [std::basic\\_ifstream< char >::basic\\_ifstream\(\)](#), and [std::basic\\_ifstream< char >::open\(\)](#).

### **open()** [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

#### **Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

### **operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

### **operator"()!**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

#### operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 __istream_type &(* __pf) (__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

#### operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

#### Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, `failbit` is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

#### operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\_ \leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [5/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                                           |                                            |
|---------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_ \leftarrow$<br>$\leftarrow$<br>$\_ \leftarrow$<br>$f$ | A variable of builtin floating point type. |
|---------------------------------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                                           |                                            |
|---------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_ \leftarrow$<br>$\leftarrow$<br>$\_ \leftarrow$<br>$f$ | A variable of builtin floating point type. |
|---------------------------------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.



**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base & (* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| <code>↵</code>  | A variable of builtin floating point type. |
| <code>_↵</code> |                                            |
| <code>↵</code>  |                                            |
| <code>_↵</code> |                                            |
| <code>f</code>  |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↵</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↵</code> | A variable of builtin integral type. |
| <code>_n</code> |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_ifstream<_CharT, _Traits>::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

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

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator<<()`.

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>_↔<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pwdow` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ifstream< _CharT, _Traits >::rdbuf () const [inline], [nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

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

**rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_ifstream< _CharT, _Traits >::`

`std::basic_ifstream< _CharT, _Traits >::seekg()`, `std::basic_ifstream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_ifstream< _CharT, _Traits >::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_ifstream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

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

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.



**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

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

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set. Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

### setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

### sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits>::pos_type std::basic_ifstream<_CharT, _Traits>::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

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

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sungetc\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

#### Parameters

|                 |                         |
|-----------------|-------------------------|
| <code>_↔</code> | The character to widen. |
| <code>_c</code> |                         |

#### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↔html>

Referenced by `std::endl()`, `std::basic_ios<char_type, traits_type>::fill()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F>` and `std::tr2::operator>>()`.

#### width() [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

#### width() [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

#### Returns

The previous value of `width()`.

#### xalloc()

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 6.237.6 Member Data Documentation

### M\_gcount

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

### adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_iostream< char_type, traits_type >::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

### basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

const `seekdir` std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

**binary**

const `openmode` std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

const `fmtflags` std::ios\_base::boolalpha [static], [inherited]

Insert/extract bool in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**cur**

const `seekdir` std::ios\_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

const `fmtflags` std::ios\_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::phiox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<>()`, and `std::phiox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>()`.

**end**

const `seekdir` std::ios\_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**eofbit**

const `iostate` std::ios\_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**failbit**

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_fstream<char>::close\(\)](#), [std::basic\\_ifstream<char>::close\(\)](#), [std::basic\\_ofstream<char>::close\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_fstream<char>::open\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::open\(\)](#), [std::basic\\_ifstream<char>::open\(\)](#), [std::basic\\_ofstream<char>::open\(\)](#), [std::basic\\_ostream<char>::open\(\)](#), [std::operator<<\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of [setf](#).

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator<<\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ws\(\)](#), and [std::ws\(\)](#).

**hex**

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::hex\(\)](#).

**in**

```
const openmode std::ios_base::in [static], [inherited]
```



Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_fstream<char>::basic_fstream()`, `std::basic_fstream<char>::basic_fstream()`, `std::basic_fstream<char>::basic_fstream()`, `std::basic_ifstream<char>::basic_ifstream()`, `std::basic_ifstream<char>::basic_ifstream()`, `std::basic_ifstream<char>::basic_ifstream()`, `std::basic_istreamstream<char>::basic_istreamstream()`, `std::basic_stringbuf<char>::basic_stringbuf()`, `std::basic_stringstream<char>::basic_stringstream()`, `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pubseekoff()`, `std::basic_streambuf<char_type, traits_type>::pubseekoff()`, `std::basic_streambuf<char_type, traits_type>::pubseekpos()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_streambuf<char_type, traits_type>::seekg()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<char>::seekoff()`, `std::basic_streambuf<char_type, traits_type>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_stringbuf<char>::seekpos()`, `std::basic_filebuf<_CharT, _Traits>::seekpos()`, `std::basic_stringbuf<char>::showmanyc()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

### internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by `std::internal()`.

### left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::left()`, and `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0>::do_put()`.

### oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by `std::oct()`.

### out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by `std::basic_fstream<char>::basic_fstream()`, `std::basic_fstream<char>::basic_fstream()`, `std::basic_fstream<char>::basic_fstream()`, `std::basic_ofstream<char>::basic_ofstream()`, `std::basic_ofstream<char>::basic_ofstream()`, `std::basic_ofstream<char>::basic_ofstream()`, `std::basic_ofstream<char>::basic_ofstream()`, `std::basic_ostreamstream<char>::basic_ostreamstream()`, `std::basic_stringbuf<char>::basic_stringbuf()`, `std::basic_stringstream<char>::basic_stringstream()`, `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::pubseekoff()`, `std::basic_streambuf<char_type, traits_type>::pubseekpos()`, `std::basic_streambuf<char_type, traits_type>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<char>::seekoff()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_streambuf<char_type, traits_type>::seekpos()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

### right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by `std::right()`.

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x1656579A, 0x4C8B4A1D >::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [fstream](#)

**6.238 \_\_gnu\_pbds::basic\_invalidation\_guarantee Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



#### 6.238.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 6.239 `std::basic_ios<_CharT, _Traits>` Class Template Reference

```
#include <basic_ios.h>
```

Inheritance diagram for `std::basic_ios<_CharT, _Traits>`:



## Public Types

- enum `erase_event` , `imbue_event` , `copyfmt_event` }
- `typedef void(* event_callback) (event __e, ios_base &__b, int __i)`
- `typedef _ios_Fmtflags fmtflags`
- `typedef _ios_losestate iostate`
- `typedef _ios_Openmode openmode`
- `typedef _ios_Seekdir seekdir`
- `typedef _CharT char_type`
- `typedef _Traits::int_type int_type`
- `typedef _Traits::pos_type pos_type`
- `typedef _Traits::off_type off_type`
- `typedef _Traits traits_type`

- typedef [ctype](#)< \_CharT > [\\_\\_ctype\\_type](#)
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_get\\_type](#)

### Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- virtual [~basic\\_ios](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- [char](#) [narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [setstate](#) ([iostate](#) \_\_state)
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \*\_\_tiestr)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [char\\_type](#) [widen](#) ([char](#) \_\_c) const
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- [operator bool](#) () const
- bool [operator!](#) () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [openmode](#) `__noreplace`
- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- [basic\\_ios](#) ()
- [basic\\_ios](#) (const [basic\\_ios](#) &)=delete
- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void [move](#) ([basic\\_ios](#) &&\_\_rhs)

- void **move** ([basic\\_ios](#) &\_\_rhs)
- [basic\\_ios](#) & **operator=** (const [basic\\_ios](#) &)=delete
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 6.239.1 Detailed Description

template<typename \_CharT, typename \_Traits>  
class std::basic\_ios< \_CharT, \_Traits >

Template class basic\_ios, virtual base class for all stream classes.

### Template Parameters

|                         |                                                                              |
|-------------------------|------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                    |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> <_CharT>. |

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

### 6.239.2 Member Typedef Documentation

#### [\\_\\_ctype\\_type](#)

```
template<typename _CharT, typename _Traits>
typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type
```

These are non-standard types.

#### [\\_\\_num\\_get\\_type](#)

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type
```

These are non-standard types.

### **\_\_num\_put\_type**

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type
```

These are non-standard types.

### **char\_type**

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### **event\_callback**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### **Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

### **fmtflags**

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`



- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

### `int_type`

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

### `off_type`

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios<_CharT, _Traits>::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### `openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`

- binary
- in
- out
- trunc

### pos\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

### seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

*\_Ios\_Seekdir* is implementation-defined. Defined values of type *seekdir* are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

### traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

## 6.239.3 Member Enumeration Documentation

### event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

## 6.239.4 Constructor & Destructor Documentation

### basic\_ios() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios (
 basic_streambuf< _CharT, _Traits > * __sb) [inline], [explicit]
```

Constructor performs initialization.

The parameter is passed by derived streams.

Referenced by [copyfmt\(\)](#).

**~basic\_ios()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_ios< _CharT, _Traits >::~~basic_ios () [inline], [virtual]
```

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

**basic\_ios()** [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios () [inline], [protected]
```

Empty.

The default constructor does nothing and is not normally accessible to users.

**6.239.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), and [std::time\\_put< \\_CharT, \\_Outlter >::put\(\)](#).

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#).

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit)
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by `std::basic_ios< char_type, traits_type >::exceptions()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::__detail::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type, std::ios_base::out >::seekg()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

### copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs)
```

Copies fields of `__rhs` into this.

### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

### eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard]
```

Fast error checking.

### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

### exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard]
```

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by `clear()`, and `copyfmt()`.

### exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline]
```

Throwing exceptions on errors.

### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard]
```

Fast error checking.

### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard]
```

Retrieves the *empty* character.

### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator bool\(\)](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and  $Q < P$ . It defaults to a space (' ') in the current locale.

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

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::chrono::operator<<()`, `std::philoX_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::philoX_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::do_get()`, `std::basic_ios<_CharT, _Traits>::do_put()`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.



Access to integer array.

#### Parameters

|                         |                       |
|-------------------------|-----------------------|
| <code>_↔<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

#### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

Referenced by [iword\(\)](#).

#### `narrow()`

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline]
```

Squeezes characters.

#### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

#### `operator bool()`

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### `operator"!()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.



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

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

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

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [imbue\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

**rddbuf() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rddbuf (
 basic_streambuf<_CharT, _Traits> * __sb)
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rddbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rddbuf(p); // ios == basic_ios<char>
```

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

**rdstate()**

```
template<typename _CharT, typename _Traits>
ios_state std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [nodiscard]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::bad\(\)](#), [clear\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::eof\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::fail\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::good\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::sync\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

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

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set. Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf() [2/2]**

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

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

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `copyfmt()`.

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
| <code>__C</code> |                         |

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios< char_type, traits_type >::fill()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F` and `std::tr2::operator>>()`.

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

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.239.6 Member Data Documentation****adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~sentry\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_iostream< char\\_type, traits\\_type >::operator<<\(\)](#), [std::basic\\_iostream< char\\_type, traits\\_type >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_iostream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Referenced by [std::dec\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekpos\(\)](#).

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract bool in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<>](#) and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, char\\_traits< \\_CharT > >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_fstream< char >::close\(\)](#), [std::basic\\_ifstream< char >::close\(\)](#), [std::basic\\_ofstream< char >::close\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).



**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::num\_get< \_CharT, \_InIter >::do\_get(),  
std::time\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< \_CharT, \_Traits >::flush(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::get(), std::basic\_istream< \_CharT, \_Traits >::get(),  
std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< \_CharT, \_Traits >::getline(), std::basic\_istream< \_CharT, \_Traits >::ignore(),  
std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_istream< \_CharT, \_Traits >::ignore(), std::basic\_iostream< char\_type, traits\_type >::operator<<(), std::basic\_iostream< char\_type, traits\_type >::operator>>(),  
std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::basic\_istream< \_CharT, \_Traits >::operator>>(), std::operator>>(),  
std::basic\_ostream< \_CharT, \_Traits >::put(), std::basic\_istream< \_CharT, \_Traits >::putback(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_iostream< char\_type, traits\_type >::seekg(), std::basic\_istream< \_CharT, \_Traits >::sync(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_istream< \_CharT, \_Traits >::ws() and std::ws().

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_istreamream< char >::basic_istreamream()`, `std::basic_stringbuf< char >::basic_stringbuf()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekpos()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< char >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::seekpos()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< char >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< char >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put< CharT, OutIter >::do_put()`, `std::left()`, and `std::phlox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0`

**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ostringstream< char >::basic\_ostringstream(), std::basic\_stringbuf< char >::basic\_stringbuf(), std::basic\_stringstream< char >::basic\_stringstream(), std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_streambuf< char\_type, traits\_type >::pubseekoff(), std::basic\_streambuf< char\_type, traits\_type >::pubseekpos(), std::basic\_streambuf< char\_type, traits\_type >::seekoff(), std::basic\_streambuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_ostringstream< \_CharT, \_Traits >::seekp(), std::basic\_streambuf< char\_type, traits\_type >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits >::seekpos(), std::basic\_stringbuf< char >::seekpos(), std::basic\_ostringstream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsputc.

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

## showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put< CharT, OutIter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

## showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by `std::noshowpos()`, and `std::showpos()`.

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x...](#) and [std::skipws\(\)](#).

### trunc

const [openmode](#) std::ios\_base::trunc [static], [inherited]  
Truncate an existing stream when opening. Default for ofstream.

### unitbuf

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]  
Flushes output after each output operation.  
Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]  
Replaces certain lowercase letters with their uppercase equivalents in generated output.  
Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
The documentation for this class was generated from the following files:

- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## 6.240 std::basic\_ostream< \_CharT, \_Traits > Class Template Reference

#include <ostream>

Inheritance diagram for std::basic\_ostream< \_CharT, \_Traits >:



### Public Types

- typedef [ctype](#)< \_CharT > \_\_ctype\_type
- typedef [ctype](#)< \_CharT > \_\_ctype\_type
- typedef [basic\\_ios](#)< \_CharT, \_Traits > \_\_ios\_type
- typedef [basic\\_ios](#)< \_CharT, \_Traits > \_\_ios\_type
- typedef [basic\\_istream](#)< \_CharT, \_Traits > \_\_istream\_type
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > \_\_num\_get\_type

- typedef `num_put`< `_CharT`, `ostreambuf_iterator`< `_CharT`, `_Traits` > > `__num_put_type`
- typedef `basic_ostream`< `_CharT`, `_Traits` > `__ostream_type`
- typedef `basic_streambuf`< `_CharT`, `_Traits` > `__streambuf_type`
- typedef `basic_streambuf`< `_CharT`, `_Traits` > `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event` , `imbue_event` , `copyfmt_event` }
- typedef void(\* `event_callback`) ( `event` \_\_e, `ios_base` & \_\_b, int \_\_i)
- typedef `_ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef `_ios_iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef `_ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_ios_Seekdir` `seekdir`
- typedef `_Traits` `traits_type`

### Public Member Functions

- `basic_iostream` ( `basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)
- virtual `~basic_iostream` ()
- template<typename `_ValueT`>  
  `basic_istream`< `_CharT`, `_Traits` > & `_M_extract` ( `_ValueT` & \_\_v)
- const `locale` & `_M_getloc` () const
- template<typename `_ValueT`>  
  `basic_ostream`< `_CharT`, `_Traits` > & `_M_insert` ( `_ValueT` \_\_v)
- void `_M_setstate` ( `iostate` \_\_state)
- bool `bad` () const
- void `clear` ( `iostate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` & \_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` ( `iostate` \_\_except)
- bool `fail` () const
- `char_type` `fill` () const
- `char_type` `fill` ( `char_type` \_\_ch)
- `fmtflags` `flags` () const
- `fmtflags` `flags` ( `fmtflags` \_\_fmtfl)
- `__ostream_type` & `flush` ()
- `streamsize` `gcount` () const
- `basic_istream`< `wchar_t` > & `getline` ( `char_type` \* \_\_s, `streamsize` \_\_n, `char_type` \_\_delim)
- `basic_istream`< `char` > & `getline` ( `char_type` \* \_\_s, `streamsize` \_\_n, `char_type` \_\_delim)
- `locale` `getloc` () const
- bool `good` () const
- `basic_istream`< `wchar_t` > & `ignore` ( `streamsize` \_\_n)
- `basic_istream`< `char` > & `ignore` ( `streamsize` \_\_n)
- `basic_istream`< `wchar_t` > & `ignore` ( `streamsize` \_\_n, `int_type` \_\_delim)
- `basic_istream`< `char` > & `ignore` ( `streamsize` \_\_n, `int_type` \_\_delim)
- `locale` `imbue` (const `locale` & \_\_loc)
- long & `inword` (int \_\_ix)
- `char` `narrow` ( `char_type` \_\_c, `char` \_\_dfault) const
- `__ostream_type` & `operator<<` ( `__streambuf_type` \* \_\_sb)

- `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (const volatile void *__p)`
  - `__ostream_type & operator<< (nullptr_t)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `__istream_type & operator>> (void *&__p)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `__ostream_type & seekp (pos_type)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`
  - `__istream_type & operator>> (long double &__f)`

## Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n, char __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

## Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

*All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.*

*If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.*

*If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.*

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static `bool sync_with_stdio (bool __sync=true)`
- static `int xalloc () throw ()`

### Static Public Attributes

- static const `openmode __noreplace`
- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`

- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

## Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

## Protected Member Functions

- **basic\_iostream** (basic\_iostream &&\_\_rhs)
- **basic\_iostream** (const basic\_iostream &)=delete
- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT>  
  \_\_istream\_type & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT>  
  \_\_ostream\_type & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- [basic\\_iostream](#) & **operator=** ([basic\\_iostream](#) &&\_\_rhs)
- [basic\\_iostream](#) & **operator=** (const [basic\\_iostream](#) &)=delete
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_iostream](#) &\_\_rhs)
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**



- [basic\\_ostream](#)< \_CharT, \_Traits > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 6.240.1 Detailed Description

template<typename [\\_CharT](#), typename [\\_Traits](#)>  
class std::basic\_iostream< [\\_CharT](#), [\\_Traits](#) >

Template class basic\_iostream.

#### Template Parameters

|                         |                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                                      |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> < <a href="#">_CharT</a> >. |

This class multiply inherits from the input and output stream classes simply to provide a single interface.

### 6.240.2 Member Typedef Documentation

#### event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <a href="#">_e</a> | One of the members of the event enum.                  |
| <a href="#">_b</a> | Reference to the ios_base object.                      |
| <a href="#">_i</a> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

#### fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

[\\_Ios\\_Fmtflags](#) is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.240.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**6.240.4 Constructor & Destructor Documentation****basic\_iostream()**

```
template<typename _CharT, typename _Traits>
std::basic_iostream< _CharT, _Traits >::basic_iostream (
 basic_streambuf< _CharT, _Traits > * __sb) [inline], [explicit]
```

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

**~basic\_iostream()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_iostream< _CharT, _Traits >::~~basic_iostream () [inline], [virtual]
```

Destructor does nothing.

**6.240.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See [std::ios\\_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::exceptions\(\)](#), [std::basic\\_fstream<char>::open\(\)](#), [std::basic\\_fstream<char>::open\(\)](#), [std::basic\\_ifstream<char>::open\(\)](#), [std::basic\\_ifstream<char>::open\(\)](#), [std::basic\\_ofstream<char>::open\(\)](#), [std::basic\\_ofstream<char>::open\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [rdbuf\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::unget\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

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

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by `clear()`, and `copyfmt()`.

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellg()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o](#)

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
```

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits`

Referenced by `std::flush()`.

**gcount()**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get() [1/6]**

```
template<typename _CharT, typename _Traits>
```

```
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets `failbit` and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`,

`std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, and `std::basic_istream< char_type, traits_type >::get()`.

**get() [2/6]**

```
template<typename _CharT, typename _Traits>
```

```
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_streambuf< _CharT, _Traits >::`

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.



**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen('\n'))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                                                                                                                       |                                               |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <a href="#"></a><br><code>__s</code> | A character array in which to store the data. |
| <a href="#"></a><br><code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s,__n,widen("\n"))`.

**getline()** [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, and `std::ios_base::goodbit`.

Referenced by `std::basic_istream< char_type, traits_type >::getline()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, `std::basic_ios< _CharT, _Traits >::do_get()`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::__detail::operator>>()`.

**ignore()** [1/4]

```
template<typename _CharT, typename _Traits>
```

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

Referenced by `std::basic_istream< char_type, traits_type >::ignore()`.

**ignore()** [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::](#)

**ignore()** [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 char __delim) [inline], [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**ignore()** [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::](#)



**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(* __pf)(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

#### **operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(* __pf)(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

#### **operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

#### **Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

#### **operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### **Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|              |                             |
|--------------|-----------------------------|
| $\leftarrow$ | A variable of pointer type. |
| $\_p$        |                             |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |



**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [9/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::endl and std::hex use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

**operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| <i>n</i>     |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outlter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<()** [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| <i>n</i>          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| <i>n</i>          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| <i>n</i>          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(* __pf) (__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or

- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

### `operator>>()` [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

### `operator>>()` [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

#### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

### `operator>>()` [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [7/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| <i>n</i>        |                                      |

**Returns**

*\*this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

**operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| <i>n</i>        |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [15/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long long & __n) [inline], [inherited]
```



Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__l</code> | A variable of builtin integral type. |
| <code>__n</code> |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `operator>>()` [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__l</code> | A variable of builtin integral type. |
| <code>__n</code> |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `operator>>()` [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__l</code> | A variable of pointer type. |
| <code>__p</code> |                             |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `peek()`

```
template<typename _CharT, typename _Traits>
__basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::endl\(\)](#), and [std::ends\(\)](#).

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| $\leftarrow$     | The character to push back into the input stream. |
| <code>__c</code> |                                                   |

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator<<()`.

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| $\leftarrow$      | Index into the array. |
| <code>__ix</code> |                       |

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

## Returns

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [imbue\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

## rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

## rdstate()

```
template<typename _CharT, typename _Traits>
ios_state std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::bad\(\)](#), [clear\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::eof\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::fail\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::good\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::sync\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#).

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters

- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

### register\_callback()

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

### seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

### seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### `seekp()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### `seekp()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf() [1/2]**

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf() [2/2]**

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.



**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `copyfmt()`.

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `__M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios< char_type, traits_type >::fill()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F` and `std::tr2::operator>>()`.

**width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width() [2/2]**

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The array to insert. |
|------------------|----------------------|

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Maximum number of characters to insert. |
| <code>_n</code> |                                         |

**Returns**

`*this`

Characters are copied from `___s` and inserted into the stream until one of the following happens:

- `___n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.240.6 Member Data Documentation****`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by [get\(\)](#), [get\(\)](#), [get\(\)](#), [get\(\)](#), [getline\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [putback\(\)](#), [read\(\)](#), [readsome\(\)](#), and [unget\(\)](#).

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::internal\(\)](#), [std::left\(\)](#), and [std::right\(\)](#).

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_set\\_buffer\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~sentry\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator<<\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by [std::dec\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekpos\(\)](#).

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits>::tellg\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<>](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, char\\_traits< \\_CharT > >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_fstream< char >::close\(\)](#), [std::basic\\_ifstream< char >::close\(\)](#), [std::basic\\_ofstream< char >::close\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::ws()`.

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_ifstream< char >::basic\_ifstream(), std::basic\_ifstream< char >::basic\_ifstream(), std::basic\_ifstream< char >::basic\_ifstream(), std::basic\_istreamream< char >::basic\_istreamream(), std::basic\_stringbuf< char >::basic\_stringbuf(), std::basic\_stringstream< char >::basic\_stringstream(), std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_ifstream< char >::open(), std::basic\_ifstream< char >::open(), std::basic\_ifstream< char >::open(), std::basic\_istreamream< \_CharT, \_Traits >::overflow(), std::basic\_filebuf< \_CharT, \_Traits >::pubseekoff(), std::basic\_streambuf< char\_type, traits\_type >::pubseekoff(), std::basic\_streambuf< char\_type, traits\_type >::pubseekpos(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_istream< \_CharT, \_Traits >::seekg(), std::basic\_streambuf< char\_type, traits\_type >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_streambuf< char\_type, traits\_type >::seekpos(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), std::basic\_stringbuf< char >::seekpos(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_stringbuf< char >::showmanyc(), std::basic\_istream< \_CharT, \_Traits >::tellg(), std::basic\_filebuf< \_CharT, \_Traits >::underflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow(), and std::basic\_filebuf< \_CharT, \_Traits >::xsgetn().

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::left()`, and `std::phlox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0`.



**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ostringstream< char >::basic\\_ostringstream\(\)](#), [std::basic\\_stringbuf< char >::basic\\_stringbuf\(\)](#), [std::basic\\_stringstream< char >::basic\\_stringstream\(\)](#), [std::basic\\_filebuf< char\\_type, traits\\_type >::M\\_set\\_buffer\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::pubseekoff\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::pubseekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< char >::seekoff\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::seekpos\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_Outiter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x...](#) and [std::skipws\(\)](#).

## trunc

const [openmode](#) std::ios\_base::trunc [static], [inherited]  
Truncate an existing stream when opening. Default for ofstream.

## unitbuf

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]  
Flushes output after each output operation.  
Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

## uppercase

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]  
Replaces certain lowercase letters with their uppercase equivalents in generated output.  
Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
The documentation for this class was generated from the following file:

- [istream](#)

## 6.241 std::basic\_istream< \_CharT, \_Traits > Class Template Reference

```
#include <istream>
```

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits >:



## Classes

- class [sentry](#)

## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
- typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
- typedef [basic\\_istream](#)< \_CharT, \_Traits > **\_\_istream\_type**
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
- typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags [fmtflags](#)
- typedef \_Traits::int\_type **int\_type**
- typedef \_ios\_istate [iostate](#)
- typedef \_Traits::off\_type **off\_type**
- typedef \_ios\_Openmode [openmode](#)
- typedef \_Traits::pos\_type **pos\_type**
- typedef \_ios\_Seekdir [seekdir](#)
- typedef \_Traits **traits\_type**
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

## Public Member Functions

- [basic\\_istream](#) (\_\_streambuf\_type \*\_\_sb)
- virtual [~basic\\_istream](#) ()
- template<typename \_ValueT>  
  [basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)< wchar\_t > & **getline** (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [basic\\_istream](#)< char > & **getline** (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n)
- [basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)

- `basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale & __loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `__istream_type & operator>> (__streambuf_type * __sb)`
  - `__istream_type & operator>> (void *& __p)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type & (* __pf) (__istream_type &))`
  - `__istream_type & operator>> (__ios_type & (* __pf) (__ios_type &))`
  - `__istream_type & operator>> (ios_base & (* __pf) (ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool & __n)`
  - `__istream_type & operator>> (short & __n)`
  - `__istream_type & operator>> (unsigned short & __n)`
  - `__istream_type & operator>> (int & __n)`
  - `__istream_type & operator>> (unsigned int & __n)`
  - `__istream_type & operator>> (long & __n)`
  - `__istream_type & operator>> (unsigned long & __n)`
  - `__istream_type & operator>> (long long & __n)`
  - `__istream_type & operator>> (unsigned long long & __n)`
- 
- `__istream_type & operator>> (float & __f)`
  - `__istream_type & operator>> (double & __f)`
  - `__istream_type & operator>> (long double & __f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
  - `__istream_type &` [get](#) (`char_type &__c`)
  - `__istream_type &` [get](#) (`char_type *__s`, [streamsize](#) `__n`, `char_type __delim`)
  - `__istream_type &` [get](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [get](#) (`__streambuf_type &__sb`, `char_type __delim`)
  - `__istream_type &` [get](#) (`__streambuf_type &__sb`)
  - `__istream_type &` [getline](#) (`char_type *__s`, [streamsize](#) `__n`, `char_type __delim`)
  - `__istream_type &` [getline](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`, `int_type __delim`)
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`)
  - `__istream_type &` [ignore](#) ()
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`, `char __delim`)
  - `int_type` [peek](#) ()
  - `__istream_type &` [read](#) (`char_type *__s`, [streamsize](#) `__n`)
  - [streamsize](#) [readsome](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [putback](#) (`char_type __c`)
  - `__istream_type &` [unget](#) ()
  - `int` [sync](#) ()
  - `pos_type` [tellg](#) ()
  - `__istream_type &` [seekg](#) (`pos_type`)
  - `__istream_type &` [seekg](#) (`off_type`, `ios_base::seekdir`)
- 
- `operator bool` () `const`
  - `bool` [operator!](#) () `const`

### Static Public Member Functions

- `static bool` [sync\\_with\\_stdio](#) (`bool __sync=true`)
- `static int` [xalloc](#) () `throw` ()

### Static Public Attributes

- `static const` [openmode](#) `__noreplace`
- `static const` [fmtflags](#) `adjustfield`
- `static const` [openmode](#) `app`
- `static const` [openmode](#) `ate`
- `static const` [iostate](#) `badbit`
- `static const` [fmtflags](#) `basefield`
- `static const` [seekdir](#) `beg`
- `static const` [openmode](#) `binary`
- `static const` [fmtflags](#) `boolalpha`
- `static const` [seekdir](#) `cur`
- `static const` [fmtflags](#) `dec`
- `static const` [seekdir](#) `end`

- static const [iosstate eofbit](#)
- static const [iosstate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iosstate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- **basic\_istream** (basic\_istream &&\_\_rhs)
- **basic\_istream** (const basic\_istream &)=delete
- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT>  
  \_\_istream\_type & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- [basic\\_istream](#) & **operator=** ([basic\\_istream](#) &&\_\_rhs)
- [basic\\_istream](#) & **operator=** (const [basic\\_istream](#) &)=delete
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)

### Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

### Friends

- class `sentry`

#### 6.241.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream<_CharT, _Traits>
```

Template class `basic_istream`.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

#### 6.241.2 Member Typedef Documentation

##### `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

**event\_callback**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

**Parameters**

|                                                                                                       |                                                        |
|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <br><code>__e</code> | One of the members of the event enum.                  |
| <br><code>__b</code> | Reference to the ios_base object.                      |
| <br><code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

**fmtflags**

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield



### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

### **seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## **6.241.3 Member Enumeration Documentation**

### **event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

#### 6.241.4 Constructor & Destructor Documentation

##### **basic\_istream()**

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
 __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#).

##### **~basic\_istream()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~~basic_istream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

#### 6.241.5 Member Function Documentation

##### **\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#), and [std::time\\_put< \\_CharT, \\_Outlter >::put\(\)](#).

##### **bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#).

##### **clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::seekg()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by `clear()`, and `copyfmt()`.

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::chrono::operator<<()`, `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void)
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

Referenced by [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), and [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim)
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
 char_type & __c)
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n) [inline]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline]
```

String extraction.



**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s, __n, widen("\n"))`.

**getline() [2/3]**

```
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline() [3/3]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, and `std::ios_base::goodbit`.

Referenced by `std::basic_istream< char_type, traits_type >::getline()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore() [1/4]**

```
template<typename _CharT, typename _Traits>
```

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 void)
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::ignore\(\)](#).

**ignore() [2/4]**

```
template<typename _CharT, typename _Traits>
```

```
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n)
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

**ignore()** [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 char __delim) [inline]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**ignore()** [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 int_type __delim)
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>`:

**imbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|



Squeezes characters.

#### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

#### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

#### `operator bool()`

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### `operator!()`

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### `operator>>()` [1/17]

```
template<typename _CharT, typename _Traits>
```

```
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
__ios_type &(* __pf) (__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

#### `operator>>()` [2/17]

```
template<typename _CharT, typename _Traits>
```

```
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
__istream_type &(* __pf) (__istream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.

For more information, see the `io manip` header.

#### `operator>>()` [3/17]

```
template<typename _CharT, typename _Traits>
```

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb)
```

Extracting into another streambuf.

#### Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

#### **operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline]
```

Integer arithmetic extractors.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### **operator>>()** [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>↵</code>   | A variable of builtin floating point type. |
| <code>__↵</code> |                                            |
| <code>↵</code>   |                                            |
| <code>__↵</code> |                                            |
| <code>f</code>   |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 float & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>↵</code>   | A variable of builtin floating point type. |
| <code>__↵</code> |                                            |
| <code>↵</code>   |                                            |
| <code>__↵</code> |                                            |
| <code>f</code>   |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [7/17]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
 int & __n)
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__↵</code> | A variable of builtin integral type. |
| <code>__n</code> |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(* __pf) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline]
```

Integer arithmetic extractors.



**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
 short & __n)
```

Integer arithmetic extractors.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::use_facet()`.

**`operator>>()` [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned int & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>_↔</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**`operator>>()` [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long & __n) [inline]
```

Integer arithmetic extractors.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `operator>>()` [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline]
```

Integer arithmetic extractors.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `operator>>()` [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline]
```

Integer arithmetic extractors.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

#### `operator>>()` [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline]
```

Basic arithmetic extractors.

### Parameters

|                                    |                             |
|------------------------------------|-----------------------------|
| <code>_↔</code><br><code>_p</code> | A variable of pointer type. |
|------------------------------------|-----------------------------|

### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

### `peek()`

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void)
```

Looking ahead in the stream.

### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

### `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
Flags access.
```

### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::chrono::operator<<()`.

### `precision()` [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

### Returns

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c)
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator<<()`.

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by [clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [imbue\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsomewhat\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::seekp\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

**rddbuf() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rddbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rddbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rddbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdstate()**

```
template<typename _CharT, typename _Traits>
ios_state std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by [std::basic\\_ios<char\\_type, traits\\_type>::bad\(\)](#), [clear\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::eof\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::fail\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::good\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_ios<char\\_type, traits\\_type>::sync\(\)](#), and [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#).

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n)
```

Extraction without delimiters.

**Parameters**

|                                                                                                    |                                        |
|----------------------------------------------------------------------------------------------------|----------------------------------------|
|  <code>__s</code> | A character array.                     |
|  <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#).

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                                                                                                      |                                        |
|------------------------------------------------------------------------------------------------------|----------------------------------------|
|  <code>__s</code> | A character array.                     |
|  <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets `eofbit` and extracts no characters
- if `A == 0`, extracts no characters

- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### register\_callback()

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

### seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir)
```

Changing the current read position.

### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

### Returns

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos)
```

Changing the current read position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf() ->pubseekpos(__pos)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

#### setf() [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

#### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

#### setf() [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

#### Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

#### Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.



**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_fstream<char>::close()`, `std::basic_ifstream<char>::close()`, `std::basic_ofstream<char>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
 void)
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void)
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `copyfmt()`.

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void)
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `__M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by [std::endl\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::fill\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::getline\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::operator<<\(\)](#), [std::operator<<\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F](#) and [std::tr2::operator>>\(\)](#).

### width() [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

#### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::operator>>\(\)](#), and [std::operator>>\(\)](#).

### width() [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

#### Returns

The previous value of `width()`.

### xalloc()

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

#### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 6.241.6 Member Data Documentation

### M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by [get\(\)](#), [get\(\)](#), [get\(\)](#), [get\(\)](#), [getline\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [ignore\(\)](#), [putback\(\)](#), [read\(\)](#), [readsome\(\)](#), and [unget\(\)](#).

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_iostream< char_type, traits_type >::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_iostream< char_type, traits_type >::seekg()`, and `std::ws()`.







**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x9E3779B9, 0x9E3779B9>::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

**6.242 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference**

```
#include <sstream>
```

Inheritance diagram for `std::basic_istream<_CharT, _Traits, _Alloc>`:



## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
  - typedef `basic_ios<_CharT, _Traits>` `__ios_type`
  - typedef `basic_istream<char_type, traits_type>` `__istream_type`
  - typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
  - typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
  - typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
  - typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
  - typedef `_Alloc` `allocator_type`
  - typedef `_CharT` `char_type`
  - enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
  - typedef void(\* `event_callback`) (`event` \_\_e, `ios_base` &\_\_b, int \_\_i)
  - typedef `_ios_Fmtflags` `fmtflags`
  - typedef `traits_type::int_type` `int_type`
  - typedef `_ios_istate` `iostate`
  - typedef `traits_type::off_type` `off_type`
  - typedef `_ios_Openmode` `openmode`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef `_ios_Seekdir` `seekdir`
  - typedef `_Traits` `traits_type`
- 
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

## Public Member Functions

- [basic\\_istream](#) ()
- [basic\\_istream](#) (\_\_string\_type && \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) (basic\_istream && \_\_rhs)
- [basic\\_istream](#) (const \_\_string\_type & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) (const basic\_istream &)=delete
- template<typename \_SAlloc>  
  [basic\\_istream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, const allocator\_type & \_\_a)
- template<typename \_SAlloc>  
  [basic\\_istream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, [ios\\_base::openmode](#) \_\_mode, const allocator\_type & \_\_a)
- template<typename \_SAlloc>  
  [basic\\_istream](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_istream](#) ([ios\\_base::openmode](#) \_\_mode)
- [basic\\_istream](#) ([ios\\_base::openmode](#) \_\_mode, const allocator\_type & \_\_a)
- [~basic\\_istream](#) ()
- template<typename \_ValueType>  
  [basic\\_istream](#)< \_CharT, \_Traits > & [\\_M\\_extract](#) (\_ValueType & \_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- [basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [basic\\_istream](#)< char > & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [basic\\_istream](#)< wchar\_t > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< char > & [ignore](#) ([streamsize](#) \_\_n)
- [basic\\_istream](#)< wchar\_t > & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- [basic\\_istream](#)< char > & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale](#) imbue (const [locale](#) & \_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- [basic\\_istream](#) & [operator=](#) ([basic\\_istream](#) && \_\_rhs)
- [basic\\_istream](#) & [operator=](#) (const [basic\\_istream](#) &)=delete
- \_\_istream\_type & [operator>>](#) (\_\_streambuf\_type \* \_\_sb)
- \_\_istream\_type & [operator>>](#) (void \*& \_\_p)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)

- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `__stringbuf_type * rdbuf () const`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `__string_type str () &&`
- `__string_type str () const &`
- `void str (__string_type && __s)`
- `void str (const __string_type & __s)`
- `template<__allocator_like _SAlloc>  
basic_string<_CharT, _Traits, _SAlloc> str (const _SAlloc & __sa) const`
- `template<__allocator_like _SAlloc>  
void str (const basic_string<_CharT, _Traits, _SAlloc> & __s)`
- `void swap (basic_istream & __rhs)`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `basic_string_view<char_type, traits_type> view () const noexcept`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`
- `__istream_type & operator>> (unsigned short & __n)`
- `__istream_type & operator>> (int & __n)`
- `__istream_type & operator>> (unsigned int & __n)`
- `__istream_type & operator>> (long & __n)`
- `__istream_type & operator>> (unsigned long & __n)`
- `__istream_type & operator>> (long long & __n)`
- `__istream_type & operator>> (unsigned long long & __n)`
- `__istream_type & operator>> (float & __f)`
- `__istream_type & operator>> (double & __f)`
- `__istream_type & operator>> (long double & __f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
  - `__istream_type &` [get](#) (`char_type &__c`)
  - `__istream_type &` [get](#) (`char_type *__s`, [streamsize](#) `__n`, `char_type __delim`)
  - `__istream_type &` [get](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [get](#) (`__streambuf_type &__sb`, `char_type __delim`)
  - `__istream_type &` [get](#) (`__streambuf_type &__sb`)
  - `__istream_type &` [getline](#) (`char_type *__s`, [streamsize](#) `__n`, `char_type __delim`)
  - `__istream_type &` [getline](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`, `int_type __delim`)
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`)
  - `__istream_type &` [ignore](#) ()
  - `__istream_type &` [ignore](#) ([streamsize](#) `__n`, `char __delim`)
  - `int_type` [peek](#) ()
  - `__istream_type &` [read](#) (`char_type *__s`, [streamsize](#) `__n`)
  - [streamsize](#) [readsome](#) (`char_type *__s`, [streamsize](#) `__n`)
  - `__istream_type &` [putback](#) (`char_type __c`)
  - `__istream_type &` [unget](#) ()
  - `int` [sync](#) ()
  - `pos_type` [tellg](#) ()
  - `__istream_type &` [seekg](#) (`pos_type`)
  - `__istream_type &` [seekg](#) (`off_type`, `ios_base::seekdir`)
- 
- [operator bool](#) () const
  - `bool` [operator!](#) () const

### Static Public Member Functions

- static `bool` [sync\\_with\\_stdio](#) (`bool __sync=true`)
- static `int` [xalloc](#) () throw ()

### Static Public Attributes

- static const [openmode](#) `__noreplace`
- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`

- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT>  
  \_\_istream\_type & [\\_M\\_extract](#) (\_ValueT &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void [move](#) ([basic\\_ios](#) &&\_\_rhs)
- void [move](#) ([basic\\_ios](#) &\_\_rhs)
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- void [swap](#) ([basic\\_ios](#) &\_\_rhs) noexcept
- void [swap](#) ([basic\\_istream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const \_\_ctype\_type \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- char\_type [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [streamsize](#) [\\_M\\_gcount](#)

- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- [const](#) [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- [const](#) [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- [int](#) [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 6.242.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istream<_CharT, _Traits, _Alloc>
```

Controlling input for std::string.

#### Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator&lt;_CharT&gt;</a> .              |

This class supports reading from objects of type std::basic\_string, using the inherited functions from std::basic\_istream. To control the associated sequence, an instance of std::basic\_stringbuf is used, which this page refers to as `sb`.

### 6.242.2 Member Typedef Documentation

#### [\\_\\_num\\_put\\_type](#)

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

#### [event\\_callback](#)

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                                          |                                       |
|------------------------------------------|---------------------------------------|
| <a href="#">_↔</a><br><a href="#">_e</a> | One of the members of the event enum. |
| <a href="#">_↔</a><br><a href="#">_b</a> | Reference to the ios_base object.     |

|                 |                                                        |
|-----------------|--------------------------------------------------------|
| <code>_i</code> | The integer provided when the callback was registered. |
|-----------------|--------------------------------------------------------|

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

### **fmtflags**

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`



### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

### **seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

## **6.242.3 Member Enumeration Documentation**

### **event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

#### 6.242.4 Constructor & Destructor Documentation

##### **basic\_istream()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using `in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

##### **basic\_istream()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

##### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::in` is automatically included in `__mode`.

Initializes `sb` using `__mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

##### **basic\_istream()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

##### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::in` is automatically included in `mode`.

Initializes `sb` using `str` and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

##### **~basic\_istream()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::~~basic_istream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

#### 6.242.5 Member Function Documentation

##### **\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::time_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios< char_type, traits_type >::exceptions()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::__detail::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::seekg()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
Throwing exceptions on errors.
```

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
```

```

f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}

```

**fail()**

```

template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.

```

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**fill()** [1/2]

```

template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.

```

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator bool\(\)](#).

**fill()** [2/2]

```

template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]

```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```

fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]

```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::chrono::operator<<()`, `std::philo_x_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::philo_x_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::basic_istream<char_type, traits_type>::get()`, and `std::basic_istream<char_type, traits_type>::get()`.

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

#### Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

#### Returns

`*this`

Returns `get(__sb, widen("\n"))`.

#### `get()` [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

#### Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

#### Returns

`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References `_M_gcount`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_streambuf<_CharT, _Traits>`:

#### `get()` [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s, __n, widen("\n"))`.

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |



**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

**getline()** [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

**getline()** [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**getline()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

#### Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

#### Returns

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References [\\_M\\_gcount](#), and [std::ios\\_base::goodbit](#).

Referenced by [std::basic\\_istream< char\\_type, traits\\_type >::getline\(\)](#).

#### getloc()

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

#### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::money\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::do\\_get\(\)](#), [std::chrono::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

#### good()

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

#### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore()** [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::ignore\(\)](#).

**ignore()** [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

**ignore()** [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 char __delim) [inline], [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

**ignore()** [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |



**iword()**

```
long & std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(* __pf)(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>() [2/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(* __pf)(__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**operator>>() [4/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

#### Parameters

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

#### operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

#### Parameters

|                                                                           |                                            |
|---------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_ \leftarrow$<br>$\leftarrow$<br>$\_ \leftarrow$<br>$f$ | A variable of builtin floating point type. |
|---------------------------------------------------------------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

#### operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

#### Parameters

|                                                                           |                                            |
|---------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_ \leftarrow$<br>$\leftarrow$<br>$\_ \leftarrow$<br>$f$ | A variable of builtin floating point type. |
|---------------------------------------------------------------------------|--------------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.



**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>()** [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**putback()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sputbackc\(\)](#).

Referenced by [std::operator<<\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <code>_ix</code> | Index into the array. |
|------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_istream< _CharT, _Traits, _Alloc >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`, `std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::](#) and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set. Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```



Setting new format flags.

### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

### setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

### str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istream< _CharT, _Traits, _Alloc >::str () const & [inline],
[nodiscard]
```

Copying out the string buffer.

### Returns

`rdbuf() -> str()`

### str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istream< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>_↔<br/>_s</code> | The string to use as a new sequence. |
|------------------------|--------------------------------------|

Calls `rdbuf() -> str(s)`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `copyfmt()`.

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References `_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::rdstate()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

Referenced by `std::__detail::operator>>()`.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Referenced by `std::endl()`, `std::basic_ios<char_type, traits_type>::fill()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F` and `std::tr2::operator>>()`.

**width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.242.6 Member Data Documentation****\_M\_gcount**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left`|`right`|`internal`. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::open\(\)](#).

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~sentry\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator<<\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::operator>>\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::readsome\(\)](#), [std::basic\\_iostream<char\\_type, traits\\_type>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::unget\(\)](#), and [std::ws\(\)](#).

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by [std::dec\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::hex\(\)](#), and [std::oct\(\)](#).

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekpos\(\)](#).

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::tellg\(\)](#), and [std::basic\\_ostream<\\_CharT, \\_Traits>::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<>](#) and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_fstream< char >::close\(\)](#), [std::basic\\_ifstream< char >::close\(\)](#), [std::basic\\_ofstream< char >::close\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::ws()`.

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_istreamstream< char >::basic_istreamstream()`, `std::basic_stringbuf< char >::basic_stringbuf()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_istreamstream< char >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekpos()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_streambuf< char_type, traits_type >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< char >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< char >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**internal**

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by [std::internal\(\)](#).

**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put< CharT, OutIter >::do_put()`, `std::left()`, and `std::phlox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0`



**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

[illegible]

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by `std::hexfloat()`, and `std::scientific()`.

## showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put< CharT, OutIter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x...](#) and [std::skipws\(\)](#).

### trunc

const [openmode](#) std::ios\_base::trunc [static], [inherited]  
Truncate an existing stream when opening. Default for ofstream.

### unitbuf

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]  
Flushes output after each output operation.  
Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]  
Replaces certain lowercase letters with their uppercase equivalents in generated output.  
Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
The documentation for this class was generated from the following file:

- [sstream](#)

## 6.243 std::basic\_ofstream< \_CharT, \_Traits > Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic\_ofstream< \_CharT, \_Traits >:



## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
  - typedef [basic\\_ostream](#)< char\_type, traits\_type > **\_\_ostream\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags [fmtflags](#)
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_istate [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**

## Public Member Functions

- [basic\\_ofstream](#) ()
- **basic\_ofstream** (basic\_ofstream && \_\_rhs)
- template<typename \_Path, typename \_Require = \_If\_fs\_path<\_Path>>>  
[basic\\_ofstream](#) (const \_Path & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- **basic\_ofstream** (const basic\_ofstream &)=delete
- [basic\\_ofstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ofstream](#) (const std::string & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- ~[basic\\_ofstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
[basic\\_ostream](#)< \_CharT, \_Traits > & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- \_\_ostream\_type & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const

- `locale imbue` (const `locale` & \_\_loc)
  - `bool is_open` ()
  - `bool is_open` () const
  - `long & iword` (int \_\_ix)
  - `char narrow` (char\_type \_\_c, char \_\_default) const
  - `template<typename _Path>`  
`_If_fs_path< _Path, void > open` (const \_Path & \_\_s, `ios_base::openmode` \_\_mode=`ios_base::out`)
  - `void open` (const char \* \_\_s, `ios_base::openmode` \_\_mode=`ios_base::out`)
  - `void open` (const `std::string` & \_\_s, `ios_base::openmode` \_\_mode=`ios_base::out`)
  - `__ostream_type & operator<<` (`__streambuf_type` \* \_\_sb)
  - `__ostream_type & operator<<` (const void \* \_\_p)
  - `__ostream_type & operator<<` (const volatile void \* \_\_p)
  - `__ostream_type & operator<<` (nullptr\_t)
  - `basic_ofstream & operator=` (`basic_ofstream` && \_\_rhs)
  - `basic_ofstream & operator=` (const `basic_ofstream` &)=delete
  - `streamsize precision` () const
  - `streamsize precision` (`streamsize` \_\_prec)
  - `void *& pword` (int \_\_ix)
  - `basic_streambuf< _CharT, _Traits > * rdbuf` (`basic_streambuf< _CharT, _Traits > * __sb`)
  - `__filebuf_type * rdbuf` () const
  - `iosstate rdstate` () const
  - `void register_callback` (`event_callback` \_\_fn, int \_\_index)
  - `__ostream_type & seekp` (off\_type, `ios_base::seekdir`)
  - `__ostream_type & seekp` (pos\_type)
  - `fmtflags setf` (`fmtflags` \_\_fmtfl)
  - `fmtflags setf` (`fmtflags` \_\_fmtfl, `fmtflags` \_\_mask)
  - `void setstate` (`iosstate` \_\_state)
  - `void swap` (`basic_ofstream` & \_\_rhs)
  - `pos_type tellp` ()
  - `basic_ostream< _CharT, _Traits > * tie` () const
  - `basic_ostream< _CharT, _Traits > * tie` (`basic_ostream< _CharT, _Traits > * __tiestr`)
  - `void unsetf` (`fmtflags` \_\_mask)
  - `char_type widen` (char \_\_c) const
  - `streamsize width` () const
  - `streamsize width` (`streamsize` \_\_wide)
- 
- `__ostream_type & operator<<` (`__ostream_type` &(\* \_\_pf)(`__ostream_type` &))
  - `__ostream_type & operator<<` (`__ios_type` &(\* \_\_pf)(`__ios_type` &))
  - `__ostream_type & operator<<` (`ios_base` &(\* \_\_pf)(`ios_base` &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<<` (long \_\_n)
- `__ostream_type & operator<<` (unsigned long \_\_n)
- `__ostream_type & operator<<` (bool \_\_n)

- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ofstream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static `bool sync_with_stdio (bool __sync=true)`
- static `int xalloc () throw ()`

### Static Public Attributes

- static const `openmode __noreplace`
- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`

- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) & \_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename [\\_ValueT](#)>  
    [\\_\\_ostream\\_type](#) & [\\_M\\_insert](#) ([\\_ValueT](#) \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) & \_\_rhs) noexcept
- void [init](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void [move](#) ([basic\\_ios](#) && \_\_rhs)
- void [move](#) ([basic\\_ios](#) & \_\_rhs)
- void [set\\_rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void [swap](#) ([basic\\_ios](#) & \_\_rhs) noexcept
- void [swap](#) ([basic\\_ostream](#) & \_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* [\\_M\\_callbacks](#)
- const [\\_\\_ctype\\_type](#) \* [\\_M\\_ctype](#)
- [iostate](#) [\\_M\\_exception](#)
- [char\\_type](#) [\\_M\\_fill](#)
- bool [\\_M\\_fill\\_init](#)
- [fmtflags](#) [\\_M\\_flags](#)
- [locale](#) [\\_M\\_ios\\_locale](#)
- [\\_Words](#) [\\_M\\_local\\_word](#) [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* [\\_M\\_num\\_get](#)
- const [\\_\\_num\\_put\\_type](#) \* [\\_M\\_num\\_put](#)
- [streamsize](#) [\\_M\\_precision](#)
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_streambuf](#)
- [iostate](#) [\\_M\\_streambuf\\_state](#)
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [\\_M\\_tie](#)
- [streamsize](#) [\\_M\\_width](#)
- [\\_Words](#) \* [\\_M\\_word](#)
- int [\\_M\\_word\\_size](#)
- [\\_Words](#) [\\_M\\_word\\_zero](#)

### 6.243.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ofstream< _CharT, _Traits >
```

Controlling output for files.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

### 6.243.2 Member Typedef Documentation

#### `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type [inherited]
```

These are non-standard types.

#### `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

#### `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`

- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### **iosstate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc



**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.243.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**6.243.4 Constructor & Destructor Documentation****basic\_ofstream() [1/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

**basic\_ofstream() [2/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.

**Parameters**

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::out` is automatically included in `__mode`.

**basic\_ofstream() [3/4]**

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | std::string specifying the filename.             |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

ios\_base::out is automatically included in `__mode`.

**basic\_ofstream() [4/4]**

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Create an output file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | filesystem::path specifying the filename.        |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

ios\_base::out is automatically included in `__mode`.

**~basic\_ofstream()**

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

**6.243.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\\_year\(\)](#), [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#), and [std::time\\_put< \\_CharT, \\_Outlter >::put\(\)](#).

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios< char_type, traits_type >::exceptions()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::__detail::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `rdbuf()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::seekg()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

**close()**

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::close () [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#).

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::phiox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits`

Referenced by `std::flush()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Tr`, `std::chrono::operator<<()`, `std::operator>>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::__detail::operator>>()`.

## imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::imbue\(\)](#), and [rdbuf\(\)](#).

Referenced by `std::chrono::operator<<()`.

## init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_istream< char_type, traits_type >::basic_istream()`, `std::basic_istream< char_type, traits_type >::basic_istream()`, `std::basic_istream< char >::basic_istream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_ostream< char_type, traits_type >::basic_ostream()`, `std::basic_ostream< char >::basic_ostream()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_stringstream< char >::basic_stringstream()`, and `std::basic_stringstream< char >::basic_stringstream()`.

## is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_ofstream< _CharT, _Traits >::is_open () [inline], [nodiscard]
```

Wrapper to test for an open file.

## Returns

```
rdbuf() -> is_open()
```

**iword()**

```
long & std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                        |                       |
|------------------------|-----------------------|
| <code>↔<br/>_ix</code> | Index into the array. |
|------------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

**narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>↔

**open() [1/3]**

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ofstream< _CharT, _Traits >::open (
 const _Path & __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__s</code> | The name of the file, as a <code>filesystem::path</code> . |
|------------------|------------------------------------------------------------|



|                     |                      |
|---------------------|----------------------|
| <code>__mode</code> | The open mode flags. |
|---------------------|----------------------|

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

#### open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

Referenced by [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), and [std::basic\\_ofstream< char >::open\(\)](#).

#### open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,mode|out)`. If that function fails, `failbit` is set in the stream's error state.

#### operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

#### operator"()!

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

### **operator<<()** [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

### **operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(* __pf) (__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

### **operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

### **Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

### **operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| $\_ \leftarrow$ | A variable of pointer type. |
| $\_p$           |                             |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

#### Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| $f$             |                                            |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `io manip` header.

#### `operator<<()` [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| $\_ \leftarrow$ | A variable of builtin integral type. |
| $\_n$           |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outiter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [16/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ofstream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

#### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::chrono::operator<<()`.

#### `precision()` [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

#### Returns

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                                  |                          |
|----------------------------------|--------------------------|
| <a href="#"><code>__c</code></a> | The character to insert. |
|----------------------------------|--------------------------|

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)  
Referenced by [std::endl\(\)](#), and [std::ends\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                                   |                       |
|-----------------------------------|-----------------------|
| <a href="#"><code>__ix</code></a> | Index into the array. |
|-----------------------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.



**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ofstream<_CharT, _Traits>::rdbuf () const [inline], [nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`,

`std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::`

`std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::`

and `std::basic_istream<_CharT, _Traits>::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <code>__fn</code> | The function to add. |
|-------------------|----------------------|

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__index</code> | The integer to pass to the function when invoked. |
|----------------------|---------------------------------------------------|

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

### **seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

#### **Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

#### **Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### **seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

#### **Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### **Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

### **setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_fstream< char >::close\(\)](#), [std::basic\\_ifstream< char >::close\(\)](#), [std::basic\\_ofstream< char >::close\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::flush\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::getline\(\)](#), [std::getline\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#), [std::operator<<\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::tr2::operator>>\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::put\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

References `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `copyfmt()`.

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios< char_type, traits_type >::fill()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD251F` and `std::tr2::operator>>()`.

**width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of width().

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `word` and `pwd` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `word` and `pwd` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `word` and `pwd` arrays.

### 6.243.6 Member Data Documentation

#### adjustfield

const `fmtflags` std::ios\_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### app

const `openmode` std::ios\_base::app [static], [inherited]

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

#### ate

const `openmode` std::ios\_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

#### badbit

const `iostate` std::ios\_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_iostream< char_type, traits_type >::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

#### basefield

const `fmtflags` std::ios\_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _Initer >::do_get()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::hex()`, and `std::oct()`.

#### beg

const `seekdir` std::ios\_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

#### binary

const `openmode` std::ios\_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_iostream< char_type, traits_type >::seekg()`, and `std::ws()`.



`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >`  
and `std::basic_ostream< _CharT, _Traits >::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

**goodbit**

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::do\\_get\\_year\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::flush\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::getline\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::operator<<\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::put\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::seekg\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::sync\(\)](#), [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::tellg\(\)](#), [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::ws\(\)](#).

## hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::hex()`.

**in**

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Referenced by `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_fstream< char >::basic_fstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_ifstream< char >::basic_ifstream()`, `std::basic_istreamstream< char >::basic_istreamstream()`, `std::basic_stringbuf< char >::basic_stringbuf()`, `std::basic_stringstream< char >::basic_stringstream()`, `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_istreamstream< char >::open()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekoff()`, `std::basic_streambuf< char_type, traits_type >::pubseekpos()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_streambuf< char_type, traits_type >::seekg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< char >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_stringbuf< char >::seekpos()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.



**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philoX\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x16FFFFF7, 0x85CBF5AF>::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [fstream](#)

**6.244 `std::basic_ostream<_CharT, _Traits>` Class Template Reference**

```
#include <ostream.h>
```

Inheritance diagram for `std::basic_ostream<_CharT, _Traits>`:



## Classes

- class [sentry](#)

## Public Types

- typedef [ctype](#)<\_CharT> **\_\_ctype\_type**
- typedef [basic\\_ios](#)<\_CharT, \_Traits> **\_\_ios\_type**
- typedef [num\\_put](#)<\_CharT, [ostreambuf\\_iterator](#)<\_CharT, \_Traits>> **\_\_num\_put\_type**
- typedef [basic\\_ostream](#)<\_CharT, \_Traits> **\_\_ostream\_type**
- typedef [basic\\_streambuf](#)<\_CharT, \_Traits> **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags **fmtflags**
- typedef \_Traits::int\_type **int\_type**
- typedef \_ios\_istate **istate**
- typedef \_Traits::off\_type **off\_type**
- typedef \_ios\_Openmode **openmode**
- typedef \_Traits::pos\_type **pos\_type**
- typedef \_ios\_Seekdir **seekdir**
- typedef \_Traits **traits\_type**
- typedef [num\\_get](#)<\_CharT, [istreambuf\\_iterator](#)<\_CharT, \_Traits>> **\_\_num\_get\_type**

## Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \*[\\_\\_sb](#))
  - virtual [~basic\\_ostream](#) ()
  - const [locale](#) & [\\_M\\_getloc](#) () const
  - template<typename [\\_ValueT](#)>  
  [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & [\\_M\\_insert](#) ([\\_ValueT](#) [\\_\\_v](#))
  - void [\\_M\\_setstate](#) ([iostate](#) [\\_\\_state](#))
  - bool [bad](#) () const
  - void [clear](#) ([iostate](#) [\\_\\_state](#)=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &[\\_\\_rhs](#))
  - bool [eof](#) () const
  - [iostate](#) [exceptions](#) () const
  - void [exceptions](#) ([iostate](#) [\\_\\_except](#))
  - bool [fail](#) () const
  - [char\\_type](#) [fill](#) () const
  - [char\\_type](#) [fill](#) ([char\\_type](#) [\\_\\_ch](#))
  - [fmtflags](#) [flags](#) () const
  - [fmtflags](#) [flags](#) ([fmtflags](#) [\\_\\_fmtfl](#))
  - [\\_\\_ostream\\_type](#) & [flush](#) ()
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [locale](#) [imbue](#) (const [locale](#) &[\\_\\_loc](#))
  - long & [iword](#) (int [\\_\\_ix](#))
  - [char](#) [narrow](#) ([char\\_type](#) [\\_\\_c](#), [char](#) [\\_\\_dfault](#)) const
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_streambuf\\_type](#) \*[\\_\\_sb](#))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (const void \*[\\_\\_p](#))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (const volatile void \*[\\_\\_p](#))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (nullptr\_t)
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) [\\_\\_prec](#))
  - void \*& [pword](#) (int [\\_\\_ix](#))
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sb](#))
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) [\\_\\_fn](#), int [\\_\\_index](#))
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([pos\\_type](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) [\\_\\_fmtfl](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) [\\_\\_fmtfl](#), [fmtflags](#) [\\_\\_mask](#))
  - void [setstate](#) ([iostate](#) [\\_\\_state](#))
  - [pos\\_type](#) [tellp](#) ()
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_tiestr](#))
  - void [unsetf](#) ([fmtflags](#) [\\_\\_mask](#))
  - [char\\_type](#) [widen](#) ([char](#) [\\_\\_c](#)) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) [\\_\\_wide](#))
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*[\\_\\_pf](#))([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*[\\_\\_pf](#))([\\_\\_ios\\_type](#) &))

- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `openmode` `__noreplace`
- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`

- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

### Protected Member Functions

- `basic_ostream` (`basic_ostream`<\_CharT, \_Traits> &)
- `basic_ostream` (`basic_ostream` &&\_\_rhs)
- `basic_ostream` (const `basic_ostream` &)=delete
- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- template<typename \_ValueT>  
    `_ostream_type` & `_M_insert` (\_ValueT \_\_v)
- void `_M_move` (`ios_base` &) noexcept
- void `_M_swap` (`ios_base` &\_\_rhs) noexcept
- void `init` (`basic_streambuf`<\_CharT, \_Traits> \*\_\_sb)
- void `move` (`basic_ios` &&\_\_rhs)
- void `move` (`basic_ios` &\_\_rhs)
- `basic_ostream` & `operator=` (`basic_ostream` &&\_\_rhs)
- `basic_ostream` & `operator=` (const `basic_ostream` &)=delete
- void `set_rdbuf` (`basic_streambuf`<\_CharT, \_Traits> \*\_\_sb)
- void `swap` (`basic_ios` &\_\_rhs) noexcept
- void `swap` (`basic_ostream` &\_\_rhs)

## Protected Attributes

- `_Callback_list` \* `_M_callbacks`
- `const __ctype_type` \* `_M_ctype`
- `iosstate` `_M_exception`
- `char_type` `_M_fill`
- `bool` `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- `const __num_get_type` \* `_M_num_get`
- `const __num_put_type` \* `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > \* `_M_streambuf`
- `iosstate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > \* `_M_tie`
- `streamsize` `_M_width`
- `_Words` \* `_M_word`
- `int` `_M_word_size`
- `_Words` `_M_word_zero`

## Friends

- class `sentry`

### 6.244.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >
```

Template class `basic_ostream`.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

### 6.244.2 Member Typedef Documentation

#### `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type [inherited]
```

These are non-standard types.



**event\_callback**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

**Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the ios_base object.                      |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

**fmtflags**

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

**iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

**openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.244.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

#### 6.244.4 Constructor & Destructor Documentation

##### basic\_ostream()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::basic_ostream (
 __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [operator<<\(\)](#).

##### ~basic\_ostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

#### 6.244.5 Member Function Documentation

##### \_M\_getloc()

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Referenced by [std::money\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_date\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_monthname\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_weekday\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\\_year\(\)](#), [std::num\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#), and [std::time\\_put<\\_CharT, \\_Outlter>::put\(\)](#).

##### bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#).

##### clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References [std::ios\\_base::badbit](#), [exceptions\(\)](#), [rdbuf\(\)](#), and [rdstate\(\)](#).

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::exceptions\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [rdbuf\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::seekg\(\)](#), and [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Referenced by `std::basic_ios<char_type, traits_type>::operator bool()`, `std::basic_ios<char_type, traits_type>::operator!()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, and `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator bool()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::chrono::operator<<()`, `std::philoX_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::philoX_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::flush()`.

Referenced by `std::flush()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::__detail::operator>>()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References `std::ios_base::getloc()`, `std::ios_base::imbue()`, and `rdbuf()`.

Referenced by `std::chrono::operator<<()`.





**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
```

```
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator!()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
```

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
__ios_type &(* __pf) (__ios_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
```

```
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
__ostream_type &(* __pf) (__ostream_type &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
```

```
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
__streambuf_type * __sb)
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

#### **operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline]
```

Integer arithmetic inserters.

#### **Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline]
```

Pointer arithmetic inserters.

#### **Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 float __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
 int __n)
```

Integer arithmetic inserters.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(* __pf) (ios_base &)) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
 short __n)
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put<\\_CharT, \\_Outiter>::put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned int __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned long __n) [inline]
```

Integer arithmetic inserters.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline]
```

Integer arithmetic inserters.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline]
```

Integer arithmetic inserters.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_↔$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c)
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

`*this`

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits](#)

Referenced by [std::endl\(\)](#), and [std::ends\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>_↔<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pwdow` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Referenced by `clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `imbue()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:



```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
References clear\(\).
```

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [nodiscard], [inherited]
Returns the error state of the stream buffer.
```

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`, `std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekp() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 off_type __off,
 ios_base::seekdir __dir)
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos)
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__mask</code> | The flags mask for <i>fmtfl</i> . |
|---------------------|-----------------------------------|

### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

### setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, `std::basic_fstream< char >::close()`, `std::basic_ifstream< char >::close()`, `std::basic_ofstream< char >::close()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream< char >::open()`, `std::basic_fstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ifstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ofstream< char >::open()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::ws()`.

### sync\_with\_stdio()

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

### tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

#### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

References [std::ios\\_base::cur](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::out](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

#### tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

#### Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

#### tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

#### Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

#### Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

#### unsetf()

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

#### Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios<char_type, traits_type>::fill()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<char_type, traits_type>::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F>` and `std::tr2::operator>>()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n)
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.244.6 Member Data Documentation****adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xspn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_iostream<char_type, traits_type>::operator<<()`, `std::basic_iostream<char_type, traits_type>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_iostream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf<_CharT, _Traits>::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<>](#) and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::sentry\(\)](#), [std::basic\\_fstream< char >::close\(\)](#), [std::basic\\_ifstream< char >::close\(\)](#), [std::basic\\_ofstream< char >::close\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_weekday\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ifstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::operator<<\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#).

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

**floatfield**

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).





**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_fstream< char >::basic\\_fstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ofstream< char >::basic\\_ofstream\(\)](#), [std::basic\\_ostringstream< char >::basic\\_ostringstream\(\)](#), [std::basic\\_stringbuf< char >::basic\\_stringbuf\(\)](#), [std::basic\\_stringstream< char >::basic\\_stringstream\(\)](#), [std::basic\\_filebuf< char\\_type, traits\\_type >::M\\_set\\_buffer\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_fstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_ofstream< char >::open\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::pubseekoff\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::pubseekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< char >::seekoff\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::seekpos\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits >::seekpos\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#).

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philoX\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x...](#), and [std::skipws\(\)](#).

### **trunc**

const [openmode](#) `std::ios_base::trunc` [static], [inherited]  
Truncate an existing stream when opening. Default for `ofstream`.

### **unitbuf**

const [fmtflags](#) `std::ios_base::unitbuf` [static], [inherited]  
Flushes output after each output operation.  
Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### **uppercase**

const [fmtflags](#) `std::ios_base::uppercase` [static], [inherited]  
Replaces certain lowercase letters with their uppercase equivalents in generated output.  
Referenced by [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).  
The documentation for this class was generated from the following files:

- [iosfwd.h](#)
- [ostream.h](#)
- [ostream.tcc](#)

## **6.245 `std::basic_ostringstream<_CharT, _Traits, _Alloc>` Class Template Reference**

```
#include <sstream>
```

Inheritance diagram for `std::basic_ostringstream<_CharT, _Traits, _Alloc>`:



## Public Types

- typedef `ctype<_CharT>` `__ctype_type`
  - typedef `basic_ios<_CharT, _Traits>` `__ios_type`
  - typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
  - typedef `basic_ostream<char_type, traits_type>` `__ostream_type`
  - typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
  - typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
  - typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`
  - typedef `_Alloc` `allocator_type`
  - typedef `_CharT` `char_type`
  - enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
  - typedef void(\* `event_callback`) (`event` \_\_e, `ios_base` &\_\_b, int \_\_i)
  - typedef `_ios_Fmtflags` `fmtflags`
  - typedef `traits_type::int_type` `int_type`
  - typedef `_ios_istate` `iostate`
  - typedef `traits_type::off_type` `off_type`
  - typedef `_ios_Openmode` `openmode`
  - typedef `traits_type::pos_type` `pos_type`
  - typedef `_ios_Seekdir` `seekdir`
  - typedef `_Traits` `traits_type`
- 
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`

## Public Member Functions

- [basic\\_ostringstream](#) ()
- [basic\\_ostringstream](#) (\_\_string\_type &&\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) (basic\_ostringstream &&\_\_rhs)
- [basic\\_ostringstream](#) (const \_\_string\_type &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) (const basic\_ostringstream &)=delete
- template<typename \_SAlloc>  
  [basic\\_ostringstream](#) (const [basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> &\_\_str, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
  [basic\\_ostringstream](#) (const [basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> &\_\_str, [ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
  [basic\\_ostringstream](#) (const [basic\\_string](#)<\_CharT, \_Traits, \_SAlloc> &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ostringstream](#) ([ios\\_base::openmode](#) \_\_mode)
- [basic\\_ostringstream](#) ([ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- ~[basic\\_ostringstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
  [basic\\_ostream](#)<\_CharT, \_Traits> & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- \_\_ostream\_type & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- \_\_ostream\_type & [operator<<](#) (\_\_streambuf\_type \*\_\_sb)
- \_\_ostream\_type & [operator<<](#) (const void \*\_\_p)
- \_\_ostream\_type & [operator<<](#) (const volatile void \*\_\_p)
- \_\_ostream\_type & [operator<<](#) (nullptr\_t)
- [basic\\_ostringstream](#) & [operator=](#) ([basic\\_ostringstream](#) &&\_\_rhs)
- [basic\\_ostringstream](#) & [operator=](#) (const [basic\\_ostringstream](#) &)=delete
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* [rdbuf](#) ([basic\\_streambuf](#)<\_CharT, \_Traits> \*\_\_sb)
- \_\_stringbuf\_type \* [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)

- `__ostream_type & seekp` (`off_type`, `ios_base::seekdir`)
  - `__ostream_type & seekp` (`pos_type`)
  - `fmtflags setf` (`fmtflags __fmtfl`)
  - `fmtflags setf` (`fmtflags __fmtfl`, `fmtflags __mask`)
  - `void setstate` (`iosstate __state`)
  - `__string_type str` () &&
  - `__string_type str` () const &
  - `void str` (`__string_type && __s`)
  - `void str` (const `__string_type & __s`)
  - `template<__allocator_like _SAlloc>`  
`basic_string<_CharT, _Traits, _SAlloc > str` (const `_SAlloc & __sa`) const
  - `template<__allocator_like _SAlloc>`  
`void str` (const `basic_string<_CharT, _Traits, _SAlloc > & __s`)
  - `void swap` (`basic_ostream & __rhs`)
  - `pos_type tellp` ()
  - `basic_ostream<_CharT, _Traits > * tie` () const
  - `basic_ostream<_CharT, _Traits > * tie` (`basic_ostream<_CharT, _Traits > * __tiestr`)
  - `void unsetf` (`fmtflags __mask`)
  - `basic_string_view< char_type, traits_type > view` () const noexcept
  - `char_type widen` (`char __c`) const
  - `streamsize width` () const
  - `streamsize width` (`streamsize __wide`)
- 
- `__ostream_type & operator<<` (`__ostream_type &(*__pf)(__ostream_type &)`)
  - `__ostream_type & operator<<` (`__ios_type &(*__pf)(__ios_type &)`)
  - `__ostream_type & operator<<` (`ios_base &(*__pf)(ios_base &)`)

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<<` (`long __n`)
  - `__ostream_type & operator<<` (`unsigned long __n`)
  - `__ostream_type & operator<<` (`bool __n`)
  - `__ostream_type & operator<<` (`short __n`)
  - `__ostream_type & operator<<` (`unsigned short __n`)
  - `__ostream_type & operator<<` (`int __n`)
  - `__ostream_type & operator<<` (`unsigned int __n`)
  - `__ostream_type & operator<<` (`long long __n`)
  - `__ostream_type & operator<<` (`unsigned long long __n`)
- 
- `__ostream_type & operator<<` (`double __f`)
  - `__ostream_type & operator<<` (`float __f`)
  - `__ostream_type & operator<<` (`long double __f`)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put(char_type __c)`
- `__ostream_type & write(const char_type *__s, streamsize __n)`
- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- static `bool sync_with_stdio(bool __sync=true)`
- static `int xalloc () throw ()`

### Static Public Attributes

- static const `openmode __noreplace`
- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iosstate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`
- static const `seekdir end`
- static const `iosstate eofbit`
- static const `iosstate failbit`
- static const `fmtflags fixed`
- static const `fmtflags floatfield`
- static const `iosstate goodbit`
- static const `fmtflags hex`
- static const `openmode in`
- static const `fmtflags internal`
- static const `fmtflags left`
- static const `fmtflags oct`
- static const `openmode out`
- static const `fmtflags right`
- static const `fmtflags scientific`
- static const `fmtflags showbase`
- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT>  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)<\_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

### 6.245.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_ostringstream< _CharT, _Traits, _Alloc >
```

Controlling output for std::string.



**Template Parameters**

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

**6.245.2 Member Typedef Documentation****`__num_get_type`**

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

**`event_callback`**

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

**Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

**`fmtflags`**

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

**seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

**6.245.3 Member Enumeration Documentation****event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

**6.245.4 Constructor & Destructor Documentation****basic\_ostringstream() [1/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

**basic\_ostringstream() [2/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::out` is automatically included in `mode`.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

**basic\_ostringstream() [3/3]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
std::basic_ostringstream<_CharT, _Traits, _Alloc>::basic_ostringstream (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using *str* and *mode*|*out*, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

**~basic\_ostringstream()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~basic_ostringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

**6.245.5 Member Function Documentation****\_M\_getloc()**

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::seekg()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

### copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

References `basic_ios()`, `std::__addressof()`, `exceptions()`, `fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `tie()`, `std::tie()`, and `std::ios_base::width()`.

### eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [nodiscard], [inherited]
```

Fast error checking.

### Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

### exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions () const [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by `clear()`, and `copyfmt()`.

**exceptions()** [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);

 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

**fail()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Referenced by `std::basic_ios< char_type, traits_type >::operator bool()`, `std::basic_ios< char_type, traits_type >::operator!()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

**fill()** [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by `copyfmt()`, and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator()`.

**fill()** [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

**flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::chrono::operator<<()`, `std::philoX_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::philoX_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::flush\(\)](#).

Referenced by [std::flush\(\)](#).

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::money\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::operator<<\(\)](#), [std::operator<<\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

**good()**

```
template<typename _CharT, typename _Traits>
```

```
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**imbue()**

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

References [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::imbue\(\)](#), and [rdbuf\(\)](#).

Referenced by [std::chrono::operator<<\(\)](#).





**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(* __pf)(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(* __pf)(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iosmanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted),  
or
- an exception occurs while getting a character from `__sb`, which sets `failbit` in the error state

If the function inserts no characters, `failbit` is set.

#### **operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### **Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

#### **Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

#### **Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [7/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__</code>  | A variable of builtin floating point type. |
| <code>__</code>  |                                            |
| <code>__</code>  |                                            |
| <code>__f</code> |                                            |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put< \\_CharT, \\_Outiter >::put\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [15/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

`*this`

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits](#)

Referenced by [std::endl\(\)](#), and [std::ends\(\)](#).

**pword()**

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.



**Parameters**

|                  |                       |
|------------------|-----------------------|
| <code>_ix</code> | Index into the array. |
|------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

**rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

**rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_ostringstream< _CharT, _Traits, _Alloc >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

**rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios< char_type, traits_type >::bad()`, `clear()`, `std::basic_ios< char_type, traits_type >::eof()`,

`std::basic_ios< char_type, traits_type >::fail()`, `std::basic_ios< char_type, traits_type >::good()`, `std::basic_istream< _CharT, _Traits >::`

`std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ios< char_type, traits_type >::`

and `std::basic_istream< _CharT, _Traits >::unget()`.

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekp() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

References `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`,

and `std::basic_ios< _CharT, _Traits >::setstate()`.

**seekp() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_fstream<char>::close()`, `std::basic_ifstream<char>::close()`, `std::basic_ofstream<char>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::sync()`, and `std::ws()`.

### `str()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream<_CharT, _Traits, _Alloc>::str () const & [inline],
[nodiscard]
```

Copying out the string buffer.

### Returns

`rdbuf ()->str ()`

### `str()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream<_CharT, _Traits, _Alloc>::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf ()->str (s)`.

### `sync_with_stdio()`

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp () [inherited]
```

Getting the current write position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, out).

References std::ios\_base::cur, std::basic\_ios<\_CharT, \_Traits>::fail(), std::ios\_base::out, and std::basic\_ios<\_CharT, \_Traits>::rdbuf()

**tie() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, std::cin is tied to std::cout.

Referenced by std::basic\_ostream<\_CharT, \_Traits>::sentry::sentry(), and copyfmt().

**tie() [2/2]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios< char_type, traits_type >::fill()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F` and `std::tr2::operator>>()`.

**width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width() [2/2]**

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of width().

**write()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

**xalloc()**

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**6.245.6 Member Data Documentation****adjustfield**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

**app**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

**ate**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

**badbit**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_iostream< char_type, traits_type >::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

**basefield**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

**beg**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.



**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::pbackfail()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_istream<_CharT, _Traits>::tellg()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<>()`, and `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>()`.

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_iostream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_iostream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_fstream<char>::close()`, `std::basic_ifstream<char>::close()`, `std::basic_ofstream<char>::close()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_iostream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::operator<<()`, `std::basic_iostream<char_type, traits_type>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_iostream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::seekp()`.

**fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by `std::fixed()`, and `std::hexfloat()`.



**left**

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::left()`, and `std::phlox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0>`.

**oct**

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Referenced by [std::oct\(\)](#).

**out**

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Referenced by std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::  
std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofs  
std::basic\_ostringstream< char >::basic\_ostringstream(), std::basic\_stringbuf< char >::basic\_stringbuf(), std::basic\_stringstream< char  
std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(),  
std::basic\_fstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(),  
std::basic\_ofstream< char >::open(), std::basic\_filebuf< \_CharT, \_Traits >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::c  
std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbckfail(), std::basic\_streambuf< char\_type, traits\_type >::pubseekoff(),  
std::basic\_streambuf< char\_type, traits\_type >::pubseekpos(), std::basic\_streambuf< char\_type, traits\_type >::seekoff(),  
std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_ostream< \_CharT, \_Traits >  
std::basic\_ostream< \_CharT, \_Traits >::seekp(), std::basic\_streambuf< char\_type, traits\_type >::seekpos(), std::basic\_stringbuf< \_Cha  
std::basic\_stringbuf< char >::seekpos(), std::basic\_ostream< \_CharT, \_Traits >::tellp(), and std::basic\_filebuf< \_CharT, \_Traits >::xsput

**right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).

Referenced by [std::right\(\)](#).

**scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

**showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by `std::num_put< CharT, OutIter >::do_put()`, `std::noshowbase()`, and `std::showbase()`.

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

### skipws

const [fmtflags](#) std::ios\_base::skipws [static], [inherited]

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x1](#) and [std::skipws\(\)](#).

### trunc

const [openmode](#) std::ios\_base::trunc [static], [inherited]

Truncate an existing stream when opening. Default for ofstream.

### unitbuf

const [fmtflags](#) std::ios\_base::unitbuf [static], [inherited]

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

### uppercase

const [fmtflags](#) std::ios\_base::uppercase [static], [inherited]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [sstream](#)

## 6.246 std::basic\_regex< \_Ch\_type, \_Rx\_traits > Class Template Reference

```
#include <regex>
```

### Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef traits\_type::locale\_type **locale\_type**
- typedef traits\_type::string\_type **string\_type**
- typedef \_Rx\_traits **traits\_type**
- typedef \_Ch\_type **value\_type**

### Public Member Functions

- [basic\\_regex](#) () noexcept
- template<typename \_FwdIter>  
[basic\\_regex](#) (\_FwdIter \_\_first, \_FwdIter \_\_last, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (basic\_regex &&\_\_rhs) noexcept=default
- [basic\\_regex](#) (const \_Ch\_type \*\_\_p, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const \_Ch\_type \*\_\_p, std::size\_t \_\_len, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const basic\_regex &\_\_rhs)=default
- template<typename \_Ch\_traits, typename \_Ch\_alloc>  
[basic\\_regex](#) (const [std::basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_s, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_f=ECMAScript)
- [~basic\\_regex](#) ()

- template<typename \_InputIterator>  
basic\_regex & assign (\_InputIterator \_\_first, \_InputIterator \_\_last, flag\_type \_\_flags=ECMAScript)
- basic\_regex & assign (basic\_regex && \_\_rhs) noexcept
- basic\_regex & assign (const \_Ch\_type \* \_\_p, flag\_type \_\_flags=ECMAScript)
- basic\_regex & assign (const \_Ch\_type \* \_\_p, size\_t \_\_len, flag\_type \_\_flags=ECMAScript)
- basic\_regex & assign (const basic\_regex & \_\_rhs) noexcept
- template<typename \_Ch\_traits, typename \_Alloc>  
basic\_regex & assign (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Alloc > & \_\_s, flag\_type \_\_flags=ECMAScript)
- basic\_regex & assign (initializer\_list< \_Ch\_type > \_\_l, flag\_type \_\_flags=ECMAScript)
- flag\_type flags () const noexcept
- locale\_type getloc () const noexcept
- locale\_type imbue (locale\_type \_\_loc)
- unsigned int mark\_count () const noexcept
- basic\_regex & operator= (basic\_regex &&) = default
- basic\_regex & operator= (const \_Ch\_type \* \_\_p)
- basic\_regex & operator= (const basic\_regex &) = default
- template<typename \_Ch\_traits, typename \_Alloc>  
basic\_regex & operator= (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Alloc > & \_\_s)
- basic\_regex & operator= (initializer\_list< \_Ch\_type > \_\_l)
- void swap (basic\_regex & \_\_rhs) noexcept

## Static Public Attributes

### Constants

*std [28.8.1](1)*

- static constexpr flag\_type **icase**
- static constexpr flag\_type **nosubs**
- static constexpr flag\_type **optimize**
- static constexpr flag\_type **collate**
- static constexpr flag\_type **ECMAScript**
- static constexpr flag\_type **basic**
- static constexpr flag\_type **extended**
- static constexpr flag\_type **awk**
- static constexpr flag\_type **grep**
- static constexpr flag\_type **egrep**
- static constexpr flag\_type **multiline**

## Friends

- template<typename \_Bp, typename \_Ap, typename \_Cp, typename \_Rp>  
bool \_\_detail::\_\_regex\_algo\_impl (\_Bp, \_Bp, match\_results< \_Bp, \_Ap > &, const basic\_regex< \_Cp, \_Rp > &, regex\_constants::match\_flag\_type, \_\_detail::\_\_RegexExecutorPolicy, bool)
- template<typename, typename, typename>  
class \_\_detail::\_\_Executor

## Related Symbols

(Note that these are not member symbols.)

- template<typename \_Ch\_type, typename \_Rx\_traits>  
void swap (basic\_regex< \_Ch\_type, \_Rx\_traits > & \_\_lhs, basic\_regex< \_Ch\_type, \_Rx\_traits > & \_\_rhs) noexcept

### 6.246.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex< _Ch_type, _Rx_traits >
```

A regular expression.

Specializations of this class template represent regular expressions constructed from sequences of character type  $\leftrightarrow$  `_Ch_type`. Use the `std::regex` typedef for `std::basic_regex<char>`.

A character sequence passed to the constructor will be parsed according to the chosen grammar, and used to create a state machine representing the regular expression. The regex object can then be passed to algorithms such as `std::regex_match` to match sequences of characters.

The `syntax_option_type` flag passed to the constructor selects from one of the supported regular expression grammars. The default is `ECMAScript` and the others are `basic`, `extended`, `awk`, `grep`, and `egrep`, which are variations on POSIX regular expressions.

Since

C++11

### 6.246.2 Constructor & Destructor Documentation

**basic\_regex()** [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex () [inline], [noexcept]
```

Constructs a basic regular expression that does not match any character sequence.

**basic\_regex()** [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const _Ch_type * __p,
 flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the sequence `[__p, __p + char_traits<_Ch_type>::length(__p))` interpreted according to the flags in `__f`.

#### Parameters

|                                       |                                                                                             |
|---------------------------------------|---------------------------------------------------------------------------------------------|
| $\leftrightarrow$<br><code>__p</code> | A pointer to the start of a C-style null-terminated string containing a regular expression. |
| $\leftrightarrow$<br><code>__f</code> | Flags indicating the syntax rules and options.                                              |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

**basic\_regex()** [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const _Ch_type * __p,
 std::size_t __len,
 flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in f.

#### Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>__p</code>   | A pointer to the start of a string containing a regular expression. |
| <code>__len</code> | The length of the string containing the regular expression.         |
| <code>__f</code>   | Flags indicating the syntax rules and options.                      |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

#### basic\_regex() [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const basic_regex< _Ch_type, _Rx_traits > & __rhs) [default]
```

Copy-constructs a basic regular expression.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

#### basic\_regex() [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 basic_regex< _Ch_type, _Rx_traits > && __rhs) [default], [noexcept]
```

Move-constructs a basic regular expression.

#### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

#### basic\_regex() [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Ch_alloc>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
```

```
const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__s</code> | A string containing a regular expression.      |
| <code>__f</code> | Flags indicating the syntax rules and options. |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

#### `basic_regex()` [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 _FwdIter __first,
 _FwdIter __last,
 flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the range `[first, last)` interpreted according to the flags in `f`.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__f</code>     | The format flags of the regular expression.                 |

#### Exceptions

|                          |                                                                      |
|--------------------------|----------------------------------------------------------------------|
| <code>regex_error</code> | if <code>[__first, __last)</code> is not a valid regular expression. |
|--------------------------|----------------------------------------------------------------------|

#### `basic_regex()` [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 initializer_list< _Ch_type > __l,
```



```
flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from an initializer list.

#### Parameters

|                                                                                            |                                             |
|--------------------------------------------------------------------------------------------|---------------------------------------------|
| <code>↩</code><br><code>__l</code><br><code>↩</code><br><code>__l</code><br><code>l</code> | The initializer list.                       |
| <code>↩</code><br><code>__l</code><br><code>↩</code><br><code>__l</code><br><code>f</code> | The format flags of the regular expression. |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

#### ~basic\_regex()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::basic_regex<_Ch_type, _Rx_traits >::~~basic_regex () [inline]
```

Destroys a basic regular expression.

### 6.246.3 Member Function Documentation

#### assign() [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
```

```
template<typename _InputIterator>
```

```
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
```

```
 _InputIterator __first,
```

```
 _InputIterator __last,
```

```
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__flags</code> | Syntax option flags.                                        |

#### Exceptions

|                          |                                                                                                                                                                                             |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex↩_error</code> is thrown, the object remains unchanged. |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
 basic_regex<_Ch_type, _Rx_traits > && __rhs) [inline], [noexcept]
```

Move-assigns one regular expression to another.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

**assign()** [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
 const _Ch_type * __p,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

**Parameters**

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style null-terminated string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                                                                   |

**Exceptions**

|                          |                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits >::assign (
 const _Ch_type * __p,
 size_t __len,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

**Parameters**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style string containing a regular expression pattern. |
| <code>__len</code>   | The length of the regular expression pattern string.                   |
| <code>__flags</code> | Syntax option flags.                                                   |

**Exceptions**

|                                    |                                                                                                                                               |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><i>regex_error</i></a> | if p does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged. |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
 const basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline], [noexcept]
```

Assigns one regular expression to another.

**Parameters**

|              |                                    |
|--------------|------------------------------------|
| <i>__rhs</i> | Another regular expression object. |
|--------------|------------------------------------|

Referenced by [std::basic\\_regex< char >::basic\\_regex\(\)](#), [std::basic\\_regex< char >::assign\(\)](#), [std::basic\\_regex< char >::operator=\(\)](#), [std::basic\\_regex< char >::operator=\(\)](#), and [std::basic\\_regex< char >::operator=\(\)](#).

**assign()** [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
 const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

**Parameters**

|                |                                                   |
|----------------|---------------------------------------------------|
| <i>__s</i>     | A string containing a regular expression pattern. |
| <i>__flags</i> | Syntax option flags.                              |

**Exceptions**

|                                    |                                                                                                                                                 |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><i>regex_error</i></a> | if __s does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged. |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|

**assign()** [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
 initializer_list< _Ch_type > __l,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

**Parameters**

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| <code>__l</code>     | An initializer list representing a regular expression. |
| <code>__flags</code> | Syntax option flags.                                   |

**Exceptions**

|                                          |                                                                                                                                                                                              |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><code>regex_error</code></a> | if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged. |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**flags()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex<_Ch_type, _Rx_traits>::flags () const [inline], [noexcept]
```

Gets the flags used to construct the regular expression or in the last call to `assign()`.

**getloc()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits>::getloc () const [inline], [noexcept]
```

Gets the locale currently imbued in the regular expression object.

**imbue()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits>::imbue (
 locale_type __loc) [inline]
```

Imbues the regular expression object with the given locale.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

**mark\_count()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex<_Ch_type, _Rx_traits>::mark_count () const [inline], [noexcept]
```

Gets the number of marked subexpressions within the regular expression.

**operator=() [1/5]**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::operator= (
 basic_regex<_Ch_type, _Rx_traits> &&) [default]
```

Move-assigns one regular expression to another.

**operator=()** [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::operator= (
 const _Ch_type * __p) [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

**Parameters**

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to the start of a null-terminated C-style string containing a regular expression. |
|------------------|---------------------------------------------------------------------------------------------|

**operator=()** [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::operator= (
 const basic_regex<_Ch_type, _Rx_traits> &) [default]
```

Assigns one regular expression to another.

**operator=()** [4/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::operator= (
 const basic_string<_Ch_type, _Ch_traits, _Alloc> & __s) [inline]
```

Replaces a regular expression with a new one constructed from a string.

**Parameters**

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__s</code> | A pointer to a string containing a regular expression. |
|------------------|--------------------------------------------------------|

**operator=()** [5/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex<_Ch_type, _Rx_traits>::operator= (
 initializer_list<_Ch_type> __l) [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

**Parameters**

|                |                       |
|----------------|-----------------------|
| <code>l</code> | The initializer list. |
|----------------|-----------------------|

**Exceptions**

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

**swap()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex< _Ch_type, _Rx_traits >::swap (
 basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline], [noexcept]
```

Swaps the contents of two regular expression objects.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

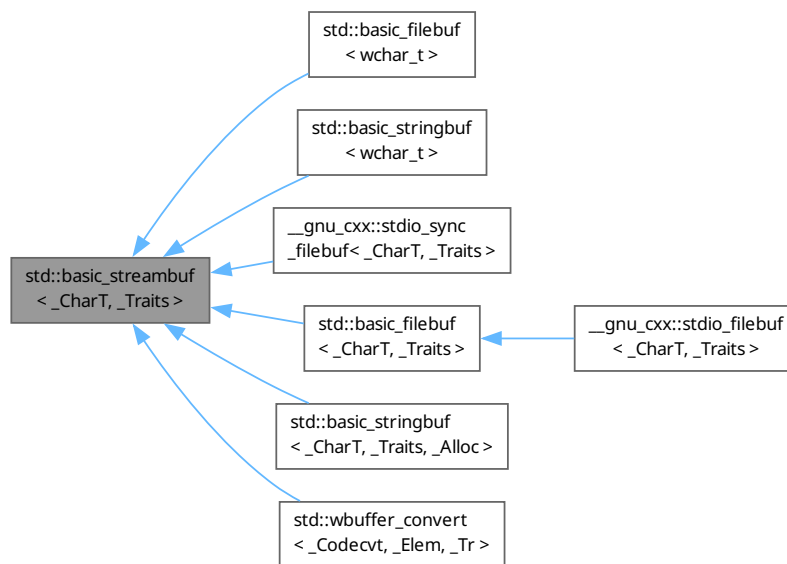
The documentation for this class was generated from the following file:

- [regex.h](#)

**6.247 std::basic\_streambuf< \_CharT, \_Traits > Class Template Reference**

```
#include <streambuf>
```

Inheritance diagram for `std::basic_streambuf< _CharT, _Traits >`:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef `traits_type::int_type` [int\\_type](#)
- typedef `traits_type::pos_type` [pos\\_type](#)
- typedef `traits_type::off_type` [off\\_type](#)
- typedef `basic_streambuf< char_type, traits_type >` [\\_\\_streambuf\\_type](#)

**Public Member Functions**

- virtual [~basic\\_streambuf](#) ()
- [locale getloc](#) () const
- [streamsize in\\_avail](#) ()
- [locale pubimbue](#) (const [locale](#) &\_\_loc)
- [int\\_type sbumpc](#) ()
- [int\\_type sgetc](#) ()
- [streamsize sgetn](#) (char\_type \*\_\_s, streamsize \_\_n)
- [int\\_type snextc](#) ()
- [int\\_type sputbackc](#) (char\_type \_\_c)
- [int\\_type sputc](#) (char\_type \_\_c)
- [streamsize sputn](#) (const char\_type \*\_\_s, streamsize \_\_n)
- [int\\_type sungetc](#) ()
- [basic\\_streambuf \\* pubsetbuf](#) (char\_type \* \_\_s, streamsize \_\_n)
- [pos\\_type pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [pos\\_type pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- [int pubsync](#) ()

**Protected Member Functions**

- [basic\\_streambuf](#) ()
- **basic\_streambuf** (const basic\_streambuf &)
- void [\\_\\_safe\\_gbump](#) (streamsize \_\_n)
- void [\\_\\_safe\\_pbump](#) (streamsize \_\_n)
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) &\_\_loc)
- [basic\\_streambuf](#) & **operator=** (const [basic\\_streambuf](#) &)
- virtual [int\\_type overflow](#) (int\_type \_\_c=traits\_type::eof())
- virtual [int\\_type pbackfail](#) (int\_type \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual [pos\\_type seekoff](#) (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual [pos\\_type seekpos](#) (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual [basic\\_streambuf< char\\_type, \\_Traits > \\* setbuf](#) (char\_type \*, streamsize)
- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual [streamsize showmanyc](#) ()
- void [swap](#) (basic\_streambuf &\_\_sb)
- virtual [int sync](#) ()
- virtual [int\\_type uflow](#) ()
- virtual [int\\_type underflow](#) ()
- virtual [streamsize xsgetn](#) (char\_type \*\_\_s, streamsize \_\_n)
- virtual [streamsize xspn](#) (const char\_type \*\_\_s, streamsize \_\_n)
- [char\\_type \\* eback](#) () const
- [char\\_type \\* gptr](#) () const
- [char\\_type \\* egptr](#) () const
- [char\\_type \\* pbase](#) () const
- [char\\_type \\* pptr](#) () const
- [char\\_type \\* ep\\_ptr](#) () const

## Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

## Friends

- `template<bool _IsMove, typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof(basic_streambuf * __sbin, basic_streambuf * __sbout, bool & __ineof)`
- `void __istream_extract(istream &, char *, streamsize)`
- `template<typename _CharT2, typename _Distance>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type advance(istreambuf_iterator< _CharT2 > &, _Distance)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2>`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type find`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc>`  
`basic_istream< _CharT2, _Traits2 > & getline(basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2,`  
`_Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc>`  
`basic_istream< _CharT2, _Traits2 > & operator>>(basic_istream< _CharT2, _Traits2 > &, basic_string<`  
`_CharT2, _Traits2, _Alloc > &)`
- `class ostreambuf_iterator< char_type, traits_type >`

### 6.247.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_streambuf< _CharT, _Traits >
```

The actual work of input and output (interface).

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:



- The controlled input sequence can be not readable.
  - The controlled output sequence can be not writable.
  - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
  - The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
- the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
- If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and *xnext* < *xend* for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

### 6.247.2 Member Typedef Documentation

#### `__streambuf_type`

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__streambuf_type
```

This is a non-standard type.

#### `char_type`

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

#### `int_type`

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf<_CharT, _Traits>::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**off\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**pos\_type**

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**6.247.3 Constructor & Destructor Documentation****~basic\_streambuf()**

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf () [inline], [virtual]
```

Destructor deallocates no buffer space.

**basic\_streambuf()**

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf () [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the `basic_streambuf` class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

**6.247.4 Member Function Documentation****eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_destroy\\_pback\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputbackc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

### egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_create\\_pback\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::in\\_avail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sgetc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_stringbuf< char\\_type, traits\\_type >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

### epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

### gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected]
```

Moving the read position.

### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

## getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline]
Locale access.
```

### Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

## gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected]
Access to the get area.
```

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

## imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
 const locale & __loc) [inline], [protected], [virtual]
Changes translations.
```

### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), and [std::basic\\_filebuf<wchar\\_t>](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::pubimbue\(\)](#).

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail () [inline]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::overflow (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<char>](#), [std::basic\\_stringbuf<wchar\\_t>](#), and [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::sputc\(\)](#), and [xsputc\(\)](#).

**pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

### Parameters

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↵</code> | The character to be inserted back into the sequence. |
| <code>_c</code> |                                                      |

### Returns

`eof()` on failure, *some other value* on success

### Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

### Note

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits>](#), [std::basic\\_stringbuf<char>](#), and [std::basic\\_stringbuf<wchar\\_t>](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::sputbackc\(\)](#), and [std::basic\\_streambuf<char\\_type, traits\\_type>::sungetc\(\)](#).

### pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase() const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits>::str\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

### pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected]
```

Moving the write position.

### Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), and [std::basic\\_streambuf<char\\_type, traits\\_type>::sputc\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_streambuf<char_t, _Traits, _Alloc>::seekoff()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_filebuf<_CharT, _Traits>::xsputn()`, and `xsputn()`.

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
pos_type __sp,
ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

#### pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

#### pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline]
```

Calls virtual `sync` function.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

#### sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline]
```

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

#### seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.



**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekoff()`.

**seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf<_CharT, _Traits>::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekpos()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf<char_type, _Traits> * std::basic_streambuf<_CharT, _Traits>::setbuf
(
 char_type * ,
 streamsize) [inline], [protected], [virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, and `std::basic_stringbuf<char>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsetbuf()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_create\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_destroy\\_pback\(\)](#), [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_set\\_buffer\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::wbuffer\\_convert\(\)](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_s](#)

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ignore\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::ign](#) and [std::basic\\_streambuf<char\\_type, traits\\_type>::snextc\(\)](#).

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
```

```
char_type * __s,
streamsize __n) [inline]
```

Entry point for xsgetn.

### Parameters

|                     |                |
|---------------------|----------------|
| $\leftarrow$<br>__s | A buffer area. |
| $\leftarrow$<br>__n | A count.       |

Returns xsgetn(\_\_s,\_\_n). The effect is to fill \_\_s[0] through \_\_s[\_\_n-1] with characters from the input sequence, if possible.

### showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc () [inline], [protected],
[virtual]
```

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to underflow() will not return traits::eof() until at least that number of characters have been supplied. If showmanyc() returns -1, then calls to underflow() or uflow() will fail. [27.5.2.4.3]/1*

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return eof() but that they will return immediately.*

The standard adds that *the morphemes of showmanyc are es-how-many-see, not show-manic.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::in_avail()`.

### snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline]
```

Getting the next character.

### Returns

The next character, or eof.

Calls sbumpc(), and if that function returns traits::eof(), so does this function. Otherwise, sgetc().

Referenced by `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, char_traits<_CharT>>::get()`, and `std::basic_istream<_CharT, char_traits<_CharT>>::readsome()`.

**sputback()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputback (
 char_type __c) [inline]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf<_CharT, _Traits>::sync () [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, and `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsync()`.

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::uflow () [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sbumpc()`, and `xsgn()`.

**underflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::underflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< wchar\\_t >](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >](#), [std::basic\\_stringbuf< char >](#), [std::basic\\_stringbuf< wchar\\_t >](#), and [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sgetc\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::uflow\(\)](#).

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

References [egptr\(\)](#), [gptr\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::xsgetn\(\)](#).

**xspn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xspn (
 const char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character insertion.

**Parameters**

|                                                                                                                    |                                        |
|--------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"> <code>__s</code></a> | A buffer area.                         |
| <a href="#"> <code>__n</code></a> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

References [pptr\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptr\(\)](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::sputn\(\)](#).

**6.247.5 Member Data Documentation****`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected]
```

Current locale setting.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::basic\\_filebuf\(\)](#).

**`_M_in_beg`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected]
```

Start of get area.

**`_M_in_cur`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected]
```

Current read area.

**`_M_in_end`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected]
```

End of get area.

**\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected]
```

Start of put area.

**\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected]
```

Current put area.

**\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected]
```

End of put area.

The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

## 6.248 **\_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator > Class Template Reference**

```
#include <string>
```

Inheritance diagram for \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >:

**Public Types**

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< typename _Base::const_iterator, basic_string >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< typename _Base::iterator, basic_string >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**



- typedef `std::reverse_iterator` < iterator > **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- **basic\_string** (`_Base` && `__base`) noexcept
- template<typename `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>`>  
**basic\_string** (`_InputIterator` `__begin`, `_InputIterator` `__end`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`basic_string` &&)=default
- **basic\_string** (`basic_string` && `__s`, const `_Allocator` & `__a`) noexcept(`std::is_nothrow_constructible<_Base, _Base, const _Allocator &>::value`)
- **basic\_string** (const `_Allocator` & `__a`) noexcept
- **basic\_string** (const `_Base` & `__base`)
- **basic\_string** (const `_CharT` \* `__s`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (const `_CharT` \* `__s`, `size_type` `__n`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (const `basic_string` &)=default
- **basic\_string** (const `basic_string` & `__s`, const `_Allocator` & `__a`)
- **basic\_string** (const `basic_string` & `__str`, `size_type` `__pos`, `size_type` `__n`=`_Base::npos`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`size_type` `__n`, `_CharT` `__c`, const `_Allocator` & `__a`=`_Allocator`())
- **basic\_string** (`std::initializer_list<_CharT>` `__l`, const `_Allocator` & `__a`=`_Allocator`())
- template<typename `_Operation`>  
constexpr void **\_\_resize\_and\_overwrite** (`size_type` `__n`, `_Operation` `__op`)
- template<typename `_Operation`>  
void **\_\_resize\_and\_overwrite** (`size_type` `__n`, `_Operation` `__op`)
- const `_Base` & **\_M\_base** () const noexcept
- `_Base` & **\_M\_base** () noexcept
- template<typename `_InputIterator`>  
`basic_string<_CharT, _Traits, _Alloc>` & **\_M\_replace\_dispatch** (`iterator` `__i1`, `iterator` `__i2`, `_InputIterator` `__k1`, `_InputIterator` `__k2`, `false_type`)
- template<typename `_InIterator`>  
`_CharT` \* **\_S\_construct** (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` & `__a`, `forward_iterator_tag`)
- template<typename `_InputIterator`>  
`basic_string` & **append** (`_InputIterator` `__first`, `_InputIterator` `__last`)
- `basic_string` & **append** (const `_CharT` \* `__s`)
- `basic_string` & **append** (const `_CharT` \* `__s`, `size_type` `__n`)
- `basic_string` & **append** (const `basic_string` & `__str`)
- `basic_string` & **append** (const `basic_string` & `__str`, `size_type` `__pos`, `size_type` `__n`)
- `basic_string` & **append** (`size_type` `__n`, `_CharT` `__c`)
- constexpr `basic_string` & **append** (const `_CharT` \* `__s`, `size_type` `__n`)
- constexpr `basic_string` & **append** (const `basic_string` & `__str`)
- constexpr `basic_string` & **append** (const `basic_string` & `__str`, `size_type` `__pos`, `size_type` `__n`=`npos`)
- constexpr `basic_string` & **append** (`initializer_list<_CharT>` `__l`)
- constexpr `basic_string` & **append** (`size_type` `__n`, `_CharT` `__c`)
- template<typename `_InputIterator`>  
`basic_string` & **assign** (`_InputIterator` `__first`, `_InputIterator` `__last`)
- `basic_string` & **assign** (`basic_string` && `__x`) noexcept(`noexcept(std::declval<_Base &>().assign(std::move(__x)))`)
- `basic_string` & **assign** (const `_CharT` \* `__s`)
- `basic_string` & **assign** (const `_CharT` \* `__s`, `size_type` `__n`)

- `basic_string & assign (const basic_string &__str, size_type __pos, size_type __n)`
- `basic_string & assign (const basic_string &__x)`
- `basic_string & assign (size_type __n, _CharT __c)`
- `basic_string & assign (std::initializer_list< _CharT > __l)`
- `constexpr basic_string & assign (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())`
- `constexpr basic_string & assign (const _CharT *__s, size_type __n)`
- `constexpr basic_string & assign (const basic_string &__str)`
- `constexpr basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)`
- `constexpr basic_string & assign (size_type __n, _CharT __c)`
- `constexpr reference at (size_type __n)`
- `reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `const_reference at (size_type __n) const`
- `constexpr reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `constexpr const_reference back () const noexcept`
- `constexpr reference back () noexcept`
- `iterator begin ()`
- `const_iterator begin () const noexcept`
- `const _CharT * c_str () const noexcept`
- `constexpr size_type capacity () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear ()`
- `constexpr int compare (const _CharT *__s) const noexcept`
- `constexpr int compare (const basic_string &__str) const`
- `constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const`
- `int compare (size_type __pos, size_type __n, const basic_string &__str) const`
- `constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const`
- `constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const`
- `int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const _CharT *__s) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const _CharT *__s, size_type __n2) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npow) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npow) const`
- `constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const`
- `constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const`
- `constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npow) const`
- `constexpr bool contains (_CharT __x) const noexcept`
- `bool contains (_CharT __x) const noexcept`
- `constexpr bool contains (basic_string_view< _CharT, _Traits > __x) const noexcept`
- `bool contains (basic_string_view< _CharT, _Traits > __x) const noexcept`
- `constexpr bool contains (const _CharT *__x) const noexcept`
- `bool contains (const _CharT *__x) const noexcept`
- `size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const`
- `constexpr size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const`
- `const_reverse_iterator crbegin () const noexcept`

- `const_reverse_iterator` **crend** () const noexcept
- `const _CharT *` **data** () const noexcept
- `constexpr _CharT *` **data** () noexcept
- `constexpr bool` **empty** () const noexcept
- `iterator` **end** ()
- `const_iterator` **end** () const noexcept
- `constexpr bool` **ends\_with** (\_CharT \_\_x) const noexcept
- `bool` **ends\_with** (\_CharT \_\_x) const noexcept
- `constexpr bool` **ends\_with** (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- `bool` **ends\_with** (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- `constexpr bool` **ends\_with** (const \_CharT \* \_\_x) const noexcept
- `bool` **ends\_with** (const \_CharT \* \_\_x) const noexcept
- `iterator` **erase** (\_\_const\_iterator \_\_first, \_\_const\_iterator \_\_last)
- `iterator` **erase** (\_\_const\_iterator \_\_position)
- `basic_string &` **erase** (size\_type \_\_pos=0, size\_type \_\_n=[Base::npos](#))
- `constexpr iterator` **erase** (\_\_const\_iterator \_\_first, \_\_const\_iterator \_\_last)
- `constexpr iterator` **erase** (\_\_const\_iterator \_\_position)
- `iterator` **erase** (iterator \_\_first, iterator \_\_last)
- `iterator` **erase** (iterator \_\_position)
- `constexpr basic_string &` **erase** (size\_type \_\_pos=0, size\_type \_\_n=[npos](#))
- `constexpr size_type` **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `size_type` **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type` **find** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `size_type` **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type` **find\_first\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `size_type` **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- `constexpr size_type` **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `constexpr size_type` **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- `size_type` **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept

- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=\_Base::npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- size\_type **find\_last\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=\_Base::npos) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- size\_type **find\_last\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **find\_last\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr allocator\_type **get\_allocator** () const noexcept
- iterator **insert** (\_const\_iterator \_\_p, \_CharT \_\_c)
- template<typename \_InputIterator>  
iterator **insert** (\_const\_iterator \_\_p, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator **insert** (const\_iterator \_\_p, std::initializer\_list<\_CharT> \_\_l)
- basic\_string & **insert** (size\_type \_\_pos, const \_CharT \* \_\_s)
- basic\_string & **insert** (size\_type \_\_pos, const \_CharT \* \_\_s, size\_type \_\_n)
- basic\_string & **insert** (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- basic\_string & **insert** (size\_type \_\_pos1, const basic\_string & \_\_str)
- basic\_string & **insert** (size\_type \_\_pos1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n)
- constexpr iterator **insert** (\_const\_iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
constexpr iterator **insert** (const\_iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- constexpr iterator **insert** (const\_iterator \_\_p, initializer\_list<\_CharT> \_\_l)

- constexpr iterator `insert` (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator `insert` (iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator>  
void `insert` (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void `insert` (iterator \_\_p, initializer\_list<\_CharT> \_\_l)
- void `insert` (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- constexpr basic\_string & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s)
- constexpr basic\_string & `insert` (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- constexpr basic\_string & `insert` (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- constexpr basic\_string & `insert` (size\_type \_\_pos1, const basic\_string & \_\_str)
- constexpr basic\_string & `insert` (size\_type \_\_pos1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n) ← `n=npos`)
- constexpr size\_type `length` () const noexcept
- constexpr size\_type `max_size` () const noexcept
- basic\_string & `operator+=` (\_CharT \_\_c)
- basic\_string & `operator+=` (const \_CharT \*\_\_s)
- basic\_string & `operator+=` (const basic\_string & \_\_str)
- basic\_string & `operator+=` (std::initializer\_list<\_CharT> \_\_l)
- constexpr basic\_string & `operator+=` (const basic\_string & \_\_str)
- basic\_string & `operator=` (\_CharT \_\_c)
- basic\_string & `operator=` (basic\_string &&) = default
- basic\_string & `operator=` (const \_CharT \*\_\_s)
- basic\_string & `operator=` (const basic\_string &) = default
- basic\_string & `operator=` (std::initializer\_list<\_CharT> \_\_l)
- reference `operator[]` (size\_type \_\_pos)
- const\_reference `operator[]` (size\_type \_\_pos) const noexcept
- constexpr reference `operator[]` (size\_type \_\_pos)
- constexpr const\_reference `operator[]` (size\_type \_\_pos) const noexcept
- void `pop_back` ()
- void `push_back` (\_CharT \_\_c)
- reverse\_iterator `rbegin` ()
- const\_reverse\_iterator `rbegin` () const noexcept
- reverse\_iterator `rend` ()
- const\_reverse\_iterator `rend` () const noexcept
- template<typename \_InputIterator>  
basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, \_InputIterator \_\_j1, \_InputIterator \_\_j2)
- basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s)
- basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const basic\_string & \_\_str)
- basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, std::initializer\_list<\_CharT> \_\_l)
- basic\_string & `replace` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- basic\_string & `replace` (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- basic\_string & `replace` (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- basic\_string & `replace` (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string & \_\_str)
- basic\_string & `replace` (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n2) ←
- constexpr basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- constexpr basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2) ←
- constexpr basic\_string & `replace` (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s)

- constexpr basic\_string & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \* \_\_s, size\_type \_\_n)
- constexpr basic\_string & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const basic\_string & \_\_str)
- constexpr basic\_string & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- constexpr basic\_string & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- constexpr basic\_string & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr basic\_string & **replace** (const\_iterator \_\_i1, const\_iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- constexpr basic\_string & **replace** (const\_iterator \_\_i1, const\_iterator \_\_i2, initializer\_list< \_CharT > \_\_l)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, \_CharT \* \_\_k1, \_CharT \* \_\_k2)
- template<class \_InputIterator>  
basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_k1, const \_CharT \* \_\_k2)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s, size\_type \_\_n)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, const basic\_string & \_\_str)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, initializer\_list< \_CharT > \_\_l)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- basic\_string & **replace** (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- constexpr basic\_string & **replace** (size\_type \_\_pos, size\_type \_\_n, const basic\_string & \_\_str)
- constexpr basic\_string & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s)
- constexpr basic\_string & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2)
- constexpr basic\_string & **replace** (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- constexpr basic\_string & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=**npos**)
- constexpr void **reserve** ()
- constexpr void **reserve** (size\_type \_\_res\_arg)
- void **reserve** (size\_type \_\_res\_arg)
- constexpr void **reserve** (size\_type \_\_res\_arg)
- void **resize** (size\_type \_\_n)
- void **resize** (size\_type \_\_n, \_CharT \_\_c)
- constexpr void **resize** (size\_type \_\_n)
- constexpr void **resize** (size\_type \_\_n, \_CharT \_\_c)
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos= **Base::npos**) const
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos=**npos**) const
- constexpr size\_type **rfind** (const basic\_string & \_\_str, size\_type \_\_pos=**npos**) const noexcept
- size\_type **rfind** (const basic\_string & \_\_str, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=**npos**) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_s, size\_type \_\_pos=**npos**) const
- constexpr size\_type **rfind** (const basic\_string & \_\_str, size\_type \_\_pos=**npos**) const noexcept
- void **shrink\_to\_fit** () noexcept
- constexpr size\_type **size** () const noexcept
- constexpr bool **starts\_with** (\_CharT \_\_x) const noexcept
- bool **starts\_with** (\_CharT \_\_x) const noexcept

- constexpr bool **starts\_with** (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- bool **starts\_with** (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- constexpr bool **starts\_with** (const \_CharT \*\_\_x) const noexcept
- bool **starts\_with** (const \_CharT \*\_\_x) const noexcept
- basic\_string **substr** (size\_type \_\_pos=0, size\_type \_\_n=[Base::npos](#)) const
- constexpr basic\_string **substr** (size\_type \_\_pos, size\_type \_\_n) &&
- constexpr basic\_string **substr** (size\_type \_\_pos=0) &&
- constexpr basic\_string **substr** (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- constexpr basic\_string\_view<\_CharT, \_Traits> **subview** (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- constexpr basic\_string\_view<\_CharT, \_Traits> **subview** (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- void **swap** (basic\_string &\_\_x) noexcept(*/\*conditional\*/*)
- constexpr void **swap** (basic\_string &\_\_s) noexcept

### Static Public Attributes

- static const size\_type [npos](#)

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::Safe\\_iterator](#)

#### 6.248.1 Detailed Description

template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>

class [\\_\\_gnu\\_debug::basic\\_string](#)<\_CharT, \_Traits, \_Allocator>

Class std::basic\_string with safety/checking/debug instrumentation.

#### 6.248.2 Member Function Documentation

##### [\\_\\_resize\\_and\\_overwrite\(\)](#) [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
 size_type __n,
 _Operation __op) [constexpr], [inherited]
```

Non-standard version of [resize\\_and\\_overwrite](#) for C++11 and above.

##### [\\_\\_resize\\_and\\_overwrite\(\)](#) [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string<_CharT, _Traits, _Alloc>::__resize_and_overwrite (
 size_type __n,
 _Operation __op) [inherited]
```

Non-standard version of [resize\\_and\\_overwrite](#) for C++11 and above.



**append()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const _CharT * __s,
 size_type __n) [inline], [constexpr], [inherited]
```

Append a C substring.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

**Returns**

Reference to this string.

**append()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by [std::basic\\_string< \\_CharT >::append\(\)](#), [std::basic\\_string< \\_CharT >::append\(\)](#), [std::basic\\_string< \\_CharT >::append\(\)](#), [std::basic\\_string< \\_CharT >::append\(\)](#), [std::\\_\\_cxx11::collate< \\_CharT >::do\\_transform\(\)](#), [std::basic\\_string< \\_CharT >::operator+=\(\)](#), [std::basic\\_string< \\_CharT >::operator+=\(\)](#), [std::basic\\_string< \\_CharT >::operator+=\(\)](#), [std::operator>>\(\)](#), [std::basic\\_string< \\_CharT, st](#) and [resize\(\)](#).

**append()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos) [inline], [constexpr], [inherited]
```

Append a substring.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|



|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
 initializer_list<_CharT> __l) [inline], [constexpr], [inherited]
```

Append an `initializer_list` of characters.

**Parameters**

|                |                                                            |
|----------------|------------------------------------------------------------|
| <code>↵</code> | The <code>initializer_list</code> of characters to append. |
| <code>↵</code> |                                                            |
| <code>↵</code> |                                                            |
| <code>↵</code> |                                                            |
| <code>/</code> |                                                            |

**Returns**

Reference to this string.

**append()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
 size_type __n,
 _CharT __c) [inline], [constexpr], [inherited]
```

Append multiple characters.

**Parameters**

|                                    |                                     |
|------------------------------------|-------------------------------------|
| <code>↵</code><br><code>__n</code> | The number of characters to append. |
| <code>↵</code><br><code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

**assign()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept],
[inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const _CharT * __s,
 size_type __n) [inline], [constexpr], [inherited]
```

Set value to a C substring.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::assign()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, `std::basic_string<_CharT>::operator=()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<char>::str()` and `std::basic_stringbuf<char>::str()`.

**assign()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos,
 size_type __n = npos) [inline], [constexpr], [inherited]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 size_type __n,
 _CharT __c) [inline], [constexpr], [inherited]
```

Set value to multiple characters.

**Parameters**

|                                  |                                 |
|----------------------------------|---------------------------------|
| <a href="#"><code>__n</code></a> | Length of the resulting string. |
| <a href="#"><code>__c</code></a> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

**at()** [1/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string<_CharT, _Traits, _Allocator >::at (
 size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <a href="#"><code>__n</code></a> | The index of the character to access. |
|----------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                   |                                        |
|-----------------------------------|----------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>n</code> is an invalid index. |
|-----------------------------------|----------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [2/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string<_CharT, _Traits, _Allocator >::at (
 size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <a href="#"><code>__n</code></a> | The index of the character to access. |
|----------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [3/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string<_CharT, _Traits, _Allocator>::at (
 size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| $\leftarrow$<br><code>__n</code> | The index of the character to access. |
|----------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [4/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string<_CharT, _Traits, _Allocator>::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                  |                                       |
|----------------------------------|---------------------------------------|
| $\leftarrow$<br><code>__n</code> | The index of the character to access. |
|----------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
 size_type __n) [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

|       |                                       |
|-------|---------------------------------------|
| $\_n$ | The index of the character to access. |
|-------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
 size_type __n) const [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

|       |                                       |
|-------|---------------------------------------|
| $\_n$ | The index of the character to access. |
|-------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**back()** [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string<_CharT, _Traits, _Allocator>::back () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

**back()** [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string<_CharT, _Traits, _Allocator>::back () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

**capacity()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::capacity () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

**compare()** [1/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
 const basic_string<_CharT, _Traits, _Allocator> & __str) const [inline], [nodiscard],
[constexpr]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Returns an integer  $< 0$  if this string is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [2/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
 size_type __pos,
 size_type __n,
 const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [3/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
 size_type __pos,
 size_type __n,
 const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.



**compare()** [4/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer  $< 0$  if the substring is ordered before `__s`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(), __s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [5/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer  $< 0$  if this substring is ordered before the string from `__s`,  $0$  if their values are equivalent, or

> 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

### compare() [6/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const [inline]
```

Compare substring against a character array.

#### Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

### compare() [7/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__pos1</code> | Index of first character of substring. |
|---------------------|----------------------------------------|

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [8/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inline]
```

Compare substring to a substring.

**Parameters**

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

**Returns**

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
```

```

 size_type __pos,
 size_type __n,
 const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline], [nodiscard],
[constexpr], [inherited]

```

Compare substring to a string.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

#### `compare()` [10/12]

```

template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const [inline], [nodiscard], [constexpr], [inherited]

```

Compare substring to a C string.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

#### `compare()` [11/12]

```

template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (

```

```

size_type __pos,
size_type __n1,
const _CharT * __s,
size_type __n2) const [inline], [nodiscard], [constexpr], [inherited]

```

Compare substring against a character array.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

#### `compare()` [12/12]

```

template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inline], [nodiscard], [constexpr], [inherited]

```

Compare substring to a substring.

#### Parameters

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

#### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`,

0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

### copy()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const [constexpr], [inherited]
```

Copy substring into C string.

### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

### Returns

Number of characters actually copied

### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

### data()

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () [inline], [nodiscard], [constexpr],
[noexcept], [inherited]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

### empty()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
bool std::basic_string< _CharT, _Traits, _Allocator >::empty () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

**erase()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
 __const_iterator __first,
 __const_iterator __last) [inline], [constexpr], [inherited]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**erase()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
 __const_iterator __position) [inline], [constexpr], [inherited]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
 iterator __first,
 iterator __last) [inherited]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**erase()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
 iterator __position) [inline], [inherited]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
 size_type __pos = 0,
 size_type __n = npos) [inline], [constexpr], [inherited]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                   |                                                       |
|-----------------------------------|-------------------------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>pos</code> is beyond the end of this string. |
|-----------------------------------|-------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are  $< __n$  characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::getline\(\)](#), [std::operator>>\(\)](#), [std::basic\\_string<\\_CharT>::pop\\_back\(\)](#), and [resize\(\)](#).



**find()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find() [4/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find() [5/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::data\(\)](#), [npos](#), and [std::size\(\)](#).

Referenced by [std::basic\\_string< \\_CharT >::find\(\)](#), [std::basic\\_string< \\_CharT >::find\(\)](#), [std::basic\\_string< \\_CharT >::find\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#).

**find()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `find()` [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `find_first_not_of()` [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [2/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a different character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of() [3/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
```

```
const basic_string< _CharT, _Traits, _Allocator > & __str,
size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_first_not_of()` [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_first_not_of()` [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a different character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_not\\_of\(\)](#).

**find\_first\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc
```

```
>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C substring.

#### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | String containing characters to locate.    |
| <code>__pos</code> | Index of character to search from.         |
| <code>__n</code>   | Number of characters from s to search for. |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### `find_first_of()` [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_first_of()` [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#).

**find\_last\_not\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of() [3/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid. |
| <code>__pos</code> | Index of character to search back from.  |
| <code>__n</code>   | Number of characters from s to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of() [4/11]**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid. |
| <code>__pos</code> | Index of character to search back from.  |
| <code>__n</code>   | Number of characters from s to consider. |



**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### **find\_last\_not\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a different character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### **find\_last\_not\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_not\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#), [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_last\\_not\\_of\(\)](#).

**find\_last\_of()** [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_last_of()` [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_last_of()` [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References `npos`, and `std::size()`.

**find\_last\_of()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string< \\_CharT >::find\\_last\\_of\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_last\\_of\(\)](#).

**front()** [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::front () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```



```
reference std::basic_string<_CharT, _Traits, _Allocator>::front () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

### get\_allocator()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

### insert() [1/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
 __const_iterator __p,
 _CharT __c) [inline], [constexpr], [inherited]
```

Insert one character.

#### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

#### Returns

Iterator referencing newly inserted char.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

### insert() [2/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
 const_iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline], [constexpr], [inherited]
```

Insert a range of characters.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 const_iterator __p,
 initializer_list< _CharT > __l) [inline], [constexpr], [inherited]
```

Insert an `initializer_list` of characters.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at.      |
| <code>__l</code> | The <code>initializer_list</code> of characters to insert. |

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

**insert()** [4/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 const_iterator __p,
 size_type __n,
 _CharT __c) [inline], [constexpr], [inherited]
```

Insert multiple characters.

**Parameters**

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                              |
| <code>__c</code> | The character to insert.                                    |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, and `std::basic_string<_CharT>::insert()`.

**insert()** [5/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
 iterator __p,
 _CharT __c) [inline], [inherited]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline], [inherited]
```

Insert a range of characters.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg,__end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [7/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 initializer_list< _CharT > __l) [inline], [inherited]
```

Insert an `initializer_list` of characters.

**Parameters**

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at.      |
| <code>__l</code> | The <code>initializer_list</code> of characters to insert. |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

**insert()** [8/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
 iterator __p,
 size_type __n,
 _CharT __c) [inline], [inherited]
```

Insert multiple characters.

#### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### `insert()` [9/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos,
 const _CharT * __s) [inline], [constexpr], [inherited]
```

Insert a C string.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
| <code>__s</code>   | The C string to insert.          |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first `n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [10/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n) [inline], [constexpr], [inherited]
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [11/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline], [constexpr], [inherited]
```

Insert multiple characters.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [12/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]
```

Insert value of a string.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [13/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos2,
 size_type __n = npos) [inline], [constexpr], [inherited]
```

Insert a substring.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <code>__pos2</code> | Start of characters in <code>str</code> to insert. |
| <code>__n</code>    | Number of characters to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                           |
|--------------------------------|---------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                           |
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::length () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::max_size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the size() of the largest possible string.

**operator+=()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.



**operator[]()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::operator\[\] (
 size_type __pos) [inline], [nodiscard], [constexpr], [inherited]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::operator\[\] (
 size_type __pos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::back\(\)](#), [std::basic\\_string<\\_CharT>::front\(\)](#), [std::basic\\_string<\\_CharT>::front\(\)](#), and [std::basic\\_string<\\_CharT>::front\(\)](#).

**replace()** [1/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 const _CharT * __s) [inline], [constexpr], [inherited]
```

Replace range of characters with C string.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |

|                                                                                                                                                                                                                        |                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"></a> <code>_s</code> | C string value to insert. |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [2/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 const _CharT * __s,
 size_type __n) [inline], [constexpr], [inherited]
```

Replace range of characters with C substring.

**Parameters**

|                                                                                                                          |                                                     |
|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| <a href="#"></a><br><code>__i1</code> | Iterator referencing start of range to replace.     |
| <a href="#"></a><br><code>__i2</code> | Iterator referencing end of range to replace.       |
| <a href="#"></a><br><code>__s</code>  | C string value to insert.                           |
| <a href="#"></a><br><code>__n</code>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [3/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
```

```

__const_iterator __i1,
__const_iterator __i2,
const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr], [inherited]

```

Replace range of characters with string.

### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

### Returns

Reference to this string.

### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### `replace()` [4/17]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 size_type __n,
 _CharT __c) [inline], [constexpr], [inherited]

```

Replace range of characters with multiple characters.

### Parameters

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↔<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↔<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↔<br/>__n</code>  | Number of characters to insert.                 |
| <code>↔<br/>__c</code>  | Character to insert.                            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 const_iterator __i1,
 const_iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline], [constexpr], [inherited]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [6/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 const_iterator __i1,
 const_iterator __i2,
 initializer_list< _CharT > __l) [inline], [constexpr], [inherited]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__l</code>  | The initializer_list of characters to insert.   |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline], [inherited]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <code><a href="#">std::length_error</a></code> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [8/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s) [inline], [inherited]
```

Replace range of characters with C string.

**Parameters**

|                                         |                                                 |
|-----------------------------------------|-------------------------------------------------|
| <code><a href="#">↵</a><br/>__i1</code> | Iterator referencing start of range to replace. |
| <code><a href="#">↵</a><br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code><a href="#">↵</a><br/>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <code><a href="#">std::length_error</a></code> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [9/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s,
 size_type __n) [inline], [inherited]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [inherited]
```

Replace range of characters with string.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 initializer_list< _CharT > __l) [inline], [inherited]
```

Replace range of characters with initializer\_list.

**Parameters**

|                                                                                                                     |                                                 |
|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"> <code>__i1</code></a> | Iterator referencing start of range to replace. |
| <a href="#"> <code>__i2</code></a> | Iterator referencing end of range to replace.   |
| <a href="#"> <code>__l</code></a>  | The initializer_list of characters to insert.   |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range [`__i1`,`__i2`). In place, characters in the range [`__k1`,`__k2`) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [12/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 size_type __n,
 _CharT __c) [inline], [inherited]
```

Replace range of characters with multiple characters.

**Parameters**

|                                                                                                                       |                                                 |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"> <code>__i1</code></a> | Iterator referencing start of range to replace. |
| <a href="#"> <code>__i2</code></a> | Iterator referencing end of range to replace.   |
| <a href="#"> <code>__n</code></a>  | Number of characters to insert.                 |



|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to insert. |
|------------------|----------------------|

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [13/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n,
 const basic_string<_CharT, _Traits, _Alloc > & __str) [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by `std::basic_string<_CharT >::append()`, `std::basic_string<_CharT >::assign()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::insert()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, `std::basic_string<_CharT >::replace()`, and `std::basic_string<_CharT >::replace()`.

**replace()** [14/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline], [constexpr], [inherited]
```

Replace characters with value of a C string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [15/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::replace
(
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) [inline], [constexpr], [inherited]
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | Index of first character to replace.             |
| <code>__n1</code>  | Number of characters to be replaced.             |
| <code>__s</code>   | C string to insert.                              |
| <code>__n2</code>  | Number of characters from <code>s</code> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos1 &gt; size()</code> .              |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [16/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos,
 size_type __n1,
 size_type __n2,
 _CharT __c) [inline], [constexpr], [inherited]
```

Replace characters with multiple characters.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> .             |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [17/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos1,
 size_type __n1,
```

```
const basic_string< _CharT, _Traits, _Alloc > & __str,
size_type __pos2,
size_type __n2 = npos) [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

### Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

### Returns

Reference to this string.

### Exceptions

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### reserve() [1/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve () [constexpr]
Equivalent to shrink_to_fit().
```

### reserve() [2/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
 size_type __res_arg) [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

### Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

### Exceptions

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

### **reserve()** [3/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string<_CharT, _Traits, _Allocator>::reserve (
 size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

#### **Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

#### **Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

### **reserve()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::reserve (
 size_type __res_arg) [constexpr], [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

#### **Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

#### **Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

References `capacity()`, `get_allocator()`, and `std::size()`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>`.

[std::tr2::operator>>\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

### resize() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
 size_type __n) [inline], [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

#### Parameters

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <a href="#">↩</a><br>__n | Number of characters the string should contain. |
|--------------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

### resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
 size_type __n,
 _CharT __c) [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

#### Parameters

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <a href="#">↩</a><br>__n | Number of characters the string should contain. |
| <a href="#">↩</a><br>__c | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to \_\_c.

References [append\(\)](#), [erase\(\)](#), and [size\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::resize\(\)](#).

### rfind() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

#### Parameters

|       |                                                       |
|-------|-------------------------------------------------------|
| __c   | Character to locate.                                  |
| __pos | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string to locate.                        |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
```



```
const basic_string<_CharT, _Traits, _Allocator> & __str,
size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `rfind()` [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `rfind()` [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**rfind()** [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::min\(\)](#), [npos](#), and [std::size\(\)](#).

**rfind()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by [std::basic\\_string<\\_CharT>::find\\_last\\_of\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), [std::basic\\_string<\\_CharT>::rfind\(\)](#), and [std::basic\\_string<\\_CharT>::rfind\(\)](#).

**size()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string<_CharT, _Traits, _Allocator>::size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**substr()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string<_CharT, _Traits, _Alloc>::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Get a substring.

**Parameters**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                   |                                     |
|-----------------------------------|-------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>__pos &gt; size()</code> . |
|-----------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

### subview() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Get a subview.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                |
| <code>__n</code>   | Number of characters in subview (default remainder). |

#### Returns

The subview.

#### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a subview using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

### subview() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr], [inherited]
```

Get a subview.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                |
| <code>__n</code>   | Number of characters in subview (default remainder). |

#### Returns

The subview.

#### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a subview using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**swap()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::swap (
 basic_string<_CharT, _Traits, _Alloc> & __s) [constexpr], [noexcept], [inherited]
```

Swap contents with another string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

References [basic\\_string\(\)](#), and [get\\_allocator\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), and [std::basic\\_string<\\_CharT>::operator=\(\)](#).

**6.248.3 Member Data Documentation****npos**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
const size_type std::basic_string<_CharT, _Traits, _Allocator>::npos [static]
```

Value returned by various member functions when they fail.

The documentation for this class was generated from the following file:

- [debug/string](#)

**6.249 `std::basic_string<_CharT, _Traits, _Alloc>` Class Template Reference**

```
#include <string>
```

Inheritance diagram for `std::basic_string< _CharT, _Traits, _Alloc >`:



## Public Types

- `typedef _Char_alloc_type allocator_type`
- `typedef _Alloc allocator_type`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, basic\_string > const_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, basic\_string > const_iterator`
- `typedef _Alloc_traits::const_pointer const_pointer`
- `typedef _CharT_alloc_traits::const_pointer const_pointer`
- `typedef _Alloc_traits::const_reference const_reference`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef _Alloc_traits::difference_type difference_type`
- `typedef _CharT_alloc_traits::difference_type difference_type`
- `typedef __gnu_cxx::__normal_iterator< pointer, basic\_string > iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, basic\_string > iterator`
- `typedef _Alloc_traits::pointer pointer`
- `typedef _CharT_alloc_traits::pointer pointer`
- `typedef _Alloc_traits::reference reference`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`

- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Alloc\_traits::size\_type **size\_type**
- typedef \_CharT\_alloc\_traits::size\_type **size\_type**
- typedef \_Traits **traits\_type**
- typedef \_Traits **traits\_type**
- typedef \_Traits::char\_type **value\_type**
- typedef \_Traits::char\_type **value\_type**

## Public Member Functions

- [basic\\_string](#) () noexcept
- constexpr [basic\\_string](#) () noexcept(*/\*conditional \*/*)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr [basic\\_string](#) (\_InputIterator \_\_beg, \_InputIterator \_\_end, const \_Alloc &\_\_a=\_Alloc())
- template<class \_InputIterator>  
[basic\\_string](#) (\_InputIterator \_\_beg, \_InputIterator \_\_end, const \_Alloc &\_\_a=\_Alloc())
- constexpr [basic\\_string](#) (basic\_string && \_\_str) noexcept
- [basic\\_string](#) (basic\_string && \_\_str) noexcept
- **basic\_string** (basic\_string && \_\_str, const \_Alloc &\_\_a)
- constexpr **basic\_string** (basic\_string && \_\_str, const \_Alloc &\_\_a) noexcept(\_Alloc\_traits::\_S\_always\_equal())
- constexpr **basic\_string** (basic\_string && \_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- constexpr **basic\_string** (basic\_string && \_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const \_Alloc &\_\_a)
- constexpr [basic\\_string](#) (const \_Alloc &\_\_a) noexcept
- template<typename = \_RequireAllocator<\_Alloc>>>  
constexpr [basic\\_string](#) (const \_CharT \* \_\_s, const \_Alloc &\_\_a=\_Alloc())
- template<typename = \_RequireAllocator<\_Alloc>>>  
[basic\\_string](#) (const \_CharT \* \_\_s, const \_Alloc &\_\_a=\_Alloc())
- constexpr [basic\\_string](#) (const \_CharT \* \_\_s, size\_type \_\_n, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const \_CharT \* \_\_s, size\_type \_\_n, const \_Alloc &\_\_a=\_Alloc())
- constexpr [basic\\_string](#) (const basic\_string & \_\_str)
- [basic\\_string](#) (const basic\_string & \_\_str)
- constexpr **basic\_string** (const basic\_string & \_\_str, const \_Alloc &\_\_a)
- **basic\_string** (const basic\_string & \_\_str, const \_Alloc &\_\_a)
- constexpr [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- constexpr [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, size\_type \_\_n)
- constexpr [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- [basic\\_string](#) (const basic\_string & \_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- constexpr [basic\\_string](#) (initializer\_list< \_CharT > \_\_l, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (initializer\_list< \_CharT > \_\_l, const \_Alloc &\_\_a=\_Alloc())
- **basic\_string** (nullptr\_t)=delete
- **basic\_string** (nullptr\_t)=delete
- template<typename = \_RequireAllocator<\_Alloc>>>  
constexpr [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- constexpr ~[basic\\_string](#) ()
- ~[basic\\_string](#) () noexcept
- template<typename \_Operation>  
constexpr void [\\_\\_resize\\_and\\_overwrite](#) (size\_type \_\_n, \_Operation \_\_op)

- `template<typename _Operation>`  
`void \_\_resize\_and\_overwrite (size_type __n, _Operation __op)`
- `template<typename _InputIterator>`  
`basic\_string< _CharT, _Traits, _Alloc > & \_M\_replace\_dispatch (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)`
- `template<typename _InIterator>`  
`_CharT * \_S\_construct (_InIterator __beg, _InIterator __end, const _Alloc & __a, forward\_iterator\_tag)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`constexpr basic\_string & append (_InputIterator __first, _InputIterator __last)`
- `template<class _InputIterator>`  
`basic\_string & append (_InputIterator __first, _InputIterator __last)`
- `constexpr basic\_string & append (const _CharT * __s)`
- `basic\_string & append (const _CharT * __s)`
- `constexpr basic\_string & append (const _CharT * __s, size_type __n)`
- `basic\_string & append (const _CharT * __s, size_type __n)`
- `constexpr basic\_string & append (const basic\_string & __str)`
- `basic\_string & append (const basic\_string & __str)`
- `constexpr basic\_string & append (const basic\_string & __str, size_type __pos, size_type __n=npos)`
- `basic\_string & append (const basic\_string & __str, size_type __pos, size_type __n=npos)`
- `constexpr basic\_string & append (initializer_list< _CharT > __l)`
- `basic\_string & append (initializer_list< _CharT > __l)`
- `constexpr basic\_string & append (size_type __n, _CharT __c)`
- `basic\_string & append (size_type __n, _CharT __c)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`constexpr basic\_string & assign (_InputIterator __first, _InputIterator __last)`
- `template<class _InputIterator>`  
`basic\_string & assign (_InputIterator __first, _InputIterator __last)`
- `constexpr basic\_string & assign (basic\_string && __str) noexcept(_Alloc_traits::_S_nothrow_move())`
- `basic\_string & assign (basic\_string && __str) noexcept(allocator\_traits< _Alloc >::is_always_equal::value)`
- `constexpr basic\_string & assign (const _CharT * __s)`
- `basic\_string & assign (const _CharT * __s)`
- `constexpr basic\_string & assign (const _CharT * __s, size_type __n)`
- `basic\_string & assign (const _CharT * __s, size_type __n)`
- `constexpr basic\_string & assign (const basic\_string & __str)`
- `basic\_string & assign (const basic\_string & __str)`
- `constexpr basic\_string & assign (const basic\_string & __str, size_type __pos, size_type __n=npos)`
- `basic\_string & assign (const basic\_string & __str, size_type __pos, size_type __n=npos)`
- `constexpr basic\_string & assign (initializer_list< _CharT > __l)`
- `basic\_string & assign (initializer_list< _CharT > __l)`
- `constexpr basic\_string & assign (size_type __n, _CharT __c)`
- `basic\_string & assign (size_type __n, _CharT __c)`
- `constexpr reference at (size_type __n)`
- `reference at (size_type __n)`
- `constexpr const_reference at (size_type __n) const`
- `const_reference at (size_type __n) const`
- `reference back ()`
- `constexpr const_reference back () const noexcept`
- `const_reference back () const noexcept`
- `constexpr reference back () noexcept`
- `iterator begin ()`
- `constexpr const_iterator begin () const noexcept`



- const\_iterator [begin](#) () const noexcept
- constexpr iterator [begin](#) () noexcept
- constexpr const \_CharT \* [c\\_str](#) () const noexcept
- const \_CharT \* [c\\_str](#) () const noexcept
- constexpr size\_type [capacity](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- constexpr const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- constexpr const\_iterator [cend](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- constexpr void [clear](#) () noexcept
- void [clear](#) () noexcept
- constexpr int [compare](#) (const \_CharT \* \_\_s) const noexcept
- int [compare](#) (const \_CharT \* \_\_s) const noexcept
- constexpr int [compare](#) (const basic\_string & \_\_str) const
- int [compare](#) (const basic\_string & \_\_str) const
- constexpr int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const basic\_string & \_\_str) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n, const basic\_string & \_\_str) const
- constexpr int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s) const
- constexpr int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2) const
- int [compare](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2) const
- constexpr int [compare](#) (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=[npos](#)) const
- int [compare](#) (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=[npos](#)) const
- constexpr bool [contains](#) (\_CharT \_\_x) const noexcept
- bool [contains](#) (\_CharT \_\_x) const noexcept
- constexpr bool [contains](#) (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- bool [contains](#) (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept
- constexpr bool [contains](#) (const \_CharT \* \_\_x) const noexcept
- bool [contains](#) (const \_CharT \* \_\_x) const noexcept
- constexpr size\_type [copy](#) (\_CharT \* \_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- size\_type [copy](#) (\_CharT \* \_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- constexpr const\_reverse\_iterator [crend](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- constexpr const \_CharT \* [data](#) () const noexcept
- const \_CharT \* [data](#) () const noexcept
- constexpr \_CharT \* [data](#) () noexcept
- \_CharT \* [data](#) () noexcept(false)
- constexpr bool [empty](#) () const noexcept
- bool [empty](#) () const noexcept
- iterator [end](#) ()
- constexpr const\_iterator [end](#) () const noexcept
- const\_iterator [end](#) () const noexcept
- constexpr iterator [end](#) () noexcept
- constexpr bool [ends\\_with](#) (\_CharT \_\_x) const noexcept
- bool [ends\\_with](#) (\_CharT \_\_x) const noexcept
- constexpr bool [ends\\_with](#) (basic\_string\_view<\_CharT, \_Traits> \_\_x) const noexcept

- bool **ends\_with** (basic\_string\_view< \_CharT, \_Traits > \_\_x) const noexcept
- constexpr bool **ends\_with** (const \_CharT \* \_\_x) const noexcept
- bool **ends\_with** (const \_CharT \* \_\_x) const noexcept
- constexpr iterator **erase** ( \_\_const\_iterator \_\_first, \_\_const\_iterator \_\_last)
- constexpr iterator **erase** ( \_\_const\_iterator \_\_position)
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_position)
- constexpr basic\_string & **erase** (size\_type \_\_pos=0, size\_type \_\_n=npos)
- basic\_string & **erase** (size\_type \_\_pos=0, size\_type \_\_n=npos)
- constexpr size\_type **find** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type **find** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- size\_type **find** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type **find** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_not\_of** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_of** ( \_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- size\_type **find\_first\_of** (const basic\_string & \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** ( \_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_not\_of** ( \_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_not\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_not\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** ( \_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_of** ( \_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_of** (const \_CharT \* \_\_s, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npo) const noexcept
- size\_type **find\_last\_of** (const basic\_string & \_\_str, size\_type \_\_pos=npo) const noexcept
- reference **front** ()
- constexpr const\_reference **front** () const noexcept

- const\_reference [front](#) () const noexcept
- constexpr reference [front](#) () noexcept
- constexpr allocator\_type [get\\_allocator](#) () const noexcept
- allocator\_type [get\\_allocator](#) () const noexcept
- constexpr iterator [insert](#) (\_\_const\_iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr iterator [insert](#) (const\_iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- constexpr iterator [insert](#) (const\_iterator \_\_p, [initializer\\_list](#)<\_CharT> \_\_l)
- constexpr iterator [insert](#) (const\_iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator [insert](#) (iterator \_\_p, \_CharT \_\_c)
- template<class \_InputIterator>  
void [insert](#) (iterator \_\_p, \_InputIterator \_\_beg, \_InputIterator \_\_end)
- void [insert](#) (iterator \_\_p, [initializer\\_list](#)<\_CharT> \_\_l)
- void [insert](#) (iterator \_\_p, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, const \_CharT \*\_\_s, size\_type \_\_n)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n↵  
n=npos)
- [basic\\_string](#) & [insert](#) (size\_type \_\_pos1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n↵  
n=npos)
- constexpr size\_type [length](#) () const noexcept
- size\_type [length](#) () const noexcept
- constexpr size\_type [max\\_size](#) () const noexcept
- size\_type [max\\_size](#) () const noexcept
- constexpr [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator+=](#) (\_CharT \_\_c)
- constexpr [basic\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator+=](#) (const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator+=](#) (const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [operator+=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- constexpr [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) (\_CharT \_\_c)
- [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&\_\_str) noexcept(*/\*conditional \*/*)
- constexpr [basic\\_string](#) & [operator=](#) ([basic\\_string](#) &&\_\_str) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- constexpr [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [operator=](#) (const \_CharT \*\_\_s)
- constexpr [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [operator=](#) (const [basic\\_string](#) &\_\_str)
- constexpr [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [operator=](#) ([initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [operator=](#) (nullptr\_t)=delete
- [basic\\_string](#) & [operator=](#) (nullptr\_t)=delete
- constexpr reference [operator\[\]](#) (size\_type \_\_pos)
- reference [operator\[\]](#) (size\_type \_\_pos)

- constexpr const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- const\_reference [operator\[\]](#) (size\_type \_\_pos) const noexcept
- void [pop\\_back](#) ()
- constexpr void [pop\\_back](#) () noexcept
- constexpr void [push\\_back](#) (\_CharT \_\_c)
- void [push\\_back](#) (\_CharT \_\_c)
- reverse\_iterator [rbegin](#) ()
- constexpr const\_reverse\_iterator [rbegin](#) () const noexcept
- const\_reverse\_iterator [rbegin](#) () const noexcept
- constexpr reverse\_iterator [rbegin](#) () noexcept
- reverse\_iterator [rend](#) ()
- constexpr const\_reverse\_iterator [rend](#) () const noexcept
- const\_reverse\_iterator [rend](#) () const noexcept
- constexpr reverse\_iterator [rend](#) () noexcept
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, \_CharT \* \_\_k1, \_CharT \* \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \* \_\_k1, const \_CharT \* \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \* \_\_s)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const [basic\\_string](#) & \_\_str)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator, typename = std::RequireInputIterator<\_InputIterator>>  
constexpr [basic\\_string](#) & [replace](#) (const\_iterator \_\_i1, const\_iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- constexpr [basic\\_string](#) & [replace](#) (const\_iterator \_\_i1, const\_iterator \_\_i2, [initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_CharT \* \_\_k1, \_CharT \* \_\_k2)
- template<class \_InputIterator>  
[basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_k1, const \_CharT \* \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const \_CharT \* \_\_s, size\_type \_\_n)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, [initializer\\_list](#)<\_CharT> \_\_l)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & [replace](#) (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) & \_\_str)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) & \_\_str)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \* \_\_s, size\_type \_\_n2)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- constexpr [basic\\_string](#) & [replace](#) (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=[npos](#))
- [basic\\_string](#) & [replace](#) (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) & \_\_str, size\_type \_\_pos2, size\_type \_\_n2=[npos](#))
- constexpr void [reserve](#) ()

- void [reserve](#) ()
- constexpr void [reserve](#) (size\_type \_\_res\_arg)
- void [reserve](#) (size\_type \_\_res\_arg)
- constexpr void [resize](#) (size\_type \_\_n)
- void [resize](#) (size\_type \_\_n)
- constexpr void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- void [resize](#) (size\_type \_\_n, \_CharT \_\_c)
- constexpr size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (\_CharT \_\_c, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const
- size\_type [rfind](#) (const \_CharT \*\_\_s, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr size\_type [rfind](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- size\_type [rfind](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=[npos](#)) const noexcept
- constexpr void [shrink\\_to\\_fit](#) () noexcept
- void [shrink\\_to\\_fit](#) () noexcept
- constexpr size\_type [size](#) () const noexcept
- size\_type [size](#) () const noexcept
- constexpr bool [starts\\_with](#) (\_CharT \_\_x) const noexcept
- bool [starts\\_with](#) (\_CharT \_\_x) const noexcept
- constexpr bool [starts\\_with](#) ([basic\\_string\\_view](#)<\_CharT, \_Traits> \_\_x) const noexcept
- bool [starts\\_with](#) ([basic\\_string\\_view](#)<\_CharT, \_Traits> \_\_x) const noexcept
- constexpr bool [starts\\_with](#) (const \_CharT \*\_\_x) const noexcept
- bool [starts\\_with](#) (const \_CharT \*\_\_x) const noexcept
- constexpr [basic\\_string](#) [substr](#) (size\_type \_\_pos, size\_type \_\_n) &&
- constexpr [basic\\_string](#) [substr](#) (size\_type \_\_pos=0) &&
- constexpr [basic\\_string](#) [substr](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- [basic\\_string](#) [substr](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- constexpr [basic\\_string\\_view](#)<\_CharT, \_Traits> [subview](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- constexpr [basic\\_string\\_view](#)<\_CharT, \_Traits> [subview](#) (size\_type \_\_pos=0, size\_type \_\_n=[npos](#)) const
- constexpr void [swap](#) ([basic\\_string](#) &\_\_s) noexcept
- void [swap](#) ([basic\\_string](#) &\_\_s) noexcept(*/\*conditional \*/*)

### Static Public Attributes

- static const size\_type [npos](#)

### Protected Types

- typedef const\_iterator \_\_[const\\_iterator](#)
- typedef iterator \_\_[const\\_iterator](#)

### Friends

- template<typename, typename, typename>  
class [basic\\_stringbuf](#)

### 6.249.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects.

Since

C++98

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

Since

C++98

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

 [_Rep]
 _M_length
[basic_string<char_type>] _M_capacity
_M_datapointer _M_refcount
_M_p -----> unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::↵M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a Copy-On-Write C++ string implementation.

### 6.249.2 Constructor & Destructor Documentation

#### basic\_string() [1/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string () [inline], [constexpr], [noexcept]
```

Default constructor creates an empty string.

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), [std::basic\\_string<\\_CharT>::assign\(\)](#), [std::basic\\_string<\\_CharT>::substr\(\)](#), and [swap\(\)](#).

#### basic\_string() [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
 const _Alloc & __a) [inline], [explicit], [constexpr], [noexcept]
```

Construct an empty string using allocator *a*.

#### basic\_string() [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr]
```

Construct string with copy of value of *\_\_str*.

#### Parameters

|              |                |
|--------------|----------------|
| <i>__str</i> | Source string. |
|--------------|----------------|

#### basic\_string() [4/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos,
 const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as copy of a substring.

#### Parameters

|              |                                        |
|--------------|----------------------------------------|
| <i>__str</i> | Source string.                         |
| <i>__pos</i> | Index of first character to copy from. |
| <i>__a</i>   | Allocator to use.                      |

#### basic\_string() [5/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos,
 size_type __n) [inline], [constexpr]
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |

**basic\_string() [6/24]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n,
 const _Alloc & __a) [inline], [constexpr]
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

**basic\_string() [7/24]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 size_type __n,
 const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string initialized by a character array.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

**basic\_string() [8/24]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
```



```
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as copy of a C string.

#### Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| $\leftarrow$<br>__s | Source C string.                                 |
| $\leftarrow$<br>__a | Allocator to use (default is default allocator). |

#### basic\_string() [9/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 size_type __n,
 _CharT __c,
 const _Alloc & __a = _Alloc()) [inline], [constexpr]
```

Construct string as multiple characters.

#### Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| $\leftarrow$<br>__n | Number of characters.                            |
| $\leftarrow$<br>__c | Character to use.                                |
| $\leftarrow$<br>__a | Allocator to use (default is default allocator). |

#### basic\_string() [10/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept]
```

Move construct string.

#### Parameters

|       |                |
|-------|----------------|
| __str | Source string. |
|-------|----------------|

The newly-created string contains the exact contents of \_\_str. \_\_str is a valid, but unspecified string.

#### basic\_string() [11/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
```

```

initializer_list< _CharT > __l,
const _Alloc & __a = _Alloc()) [inline], [constexpr]

```

Construct string from an initializer list.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | std::initializer_list of characters.             |
| <code>__a</code> | Allocator to use (default is default allocator). |

#### basic\_string() [12/24]

```

template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 _InputIterator __beg,
 _InputIterator __end,
 const _Alloc & __a = _Alloc()) [inline], [constexpr]

```

Construct string as copy of a range.

#### Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

#### ~basic\_string() [1/2]

```

template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string () [inline], [constexpr]

```

Destroy the string instance.

#### basic\_string() [13/24]

```

template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string () [inline], [noexcept]

```

Default constructor creates an empty string.

#### basic\_string() [14/24]

```

template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _Alloc & __a) [inline], [explicit]

```

Construct an empty string using allocator *a*.

**basic\_string()** [15/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Construct string with copy of value of *str*.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**basic\_string()** [16/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 const _Alloc & __a = _Alloc())
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__a</code>   | Allocator to use.                      |

**basic\_string()** [17/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n)
```

Construct string as copy of a substring.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |

**basic\_string()** [18/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
```

```
size_type __n,
const _Alloc & __a)
```

Construct string as copy of a substring.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

#### **basic\_string()** [19/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 size_type __n,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string initialized by a character array.

#### Parameters

|                        |                                                  |
|------------------------|--------------------------------------------------|
| <code>↔<br/>__s</code> | Source character array.                          |
| <code>↔<br/>__n</code> | Number of characters to copy.                    |
| <code>↔<br/>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

#### **basic\_string()** [20/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a C string.

#### Parameters

|                        |                                                  |
|------------------------|--------------------------------------------------|
| <code>↔<br/>__s</code> | Source C string.                                 |
| <code>↔<br/>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [21/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
 size_type __n,
 _CharT __c,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as multiple characters.

**Parameters**

|                                                                                                       |                                                  |
|-------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <br><code>__n</code> | Number of characters.                            |
| <br><code>__c</code> | Character to use.                                |
| <br><code>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [22/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
 basic_string<_CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move construct string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

**basic\_string()** [23/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
 initializer_list<_CharT > __l,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

**Parameters**

|                                                                                                         |                                                  |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <br><code>__l</code> | std::initializer_list of characters.             |
| <br><code>__a</code> | Allocator to use (default is default allocator). |

**basic\_string()** [24/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 _InputIterator __beg,
 _InputIterator __end,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a range.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

**~basic\_string()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string () [inline], [noexcept]
```

Destroy the string instance.

**6.249.3 Member Function Documentation****\_\_resize\_and\_overwrite()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
 size_type __n,
 _Operation __op) [constexpr]
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

**\_\_resize\_and\_overwrite()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
 size_type __n,
 _Operation __op)
```

Non-standard version of `resize_and_overwrite` for C++11 and above.

**append()** [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
 _InputIterator __first,
 _InputIterator __last) [inline], [constexpr]
```

Append a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Appends characters in the range `[__first,__last)` to this string.

**append()** [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Append a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Appends characters in the range `[__first,__last)` to this string.

**append()** [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s) [inline], [constexpr]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [4/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                                                                                          |                         |
|------------------------------------------------------------------------------------------|-------------------------|
| <br>__s | The C string to append. |
|------------------------------------------------------------------------------------------|-------------------------|

**Returns**

Reference to this string.

**append()** [5/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s,
 size_type __n) [inline], [constexpr]
```

Append a C substring.

**Parameters**

|                                                                                          |                                     |
|------------------------------------------------------------------------------------------|-------------------------------------|
| <br>__s | The C string to append.             |
| <br>__n | The number of characters to append. |

**Returns**

Reference to this string.

**append()** [6/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s,
 size_type __n)
```

Append a C substring.

**Parameters**

|                                                                                            |                                     |
|--------------------------------------------------------------------------------------------|-------------------------------------|
| <br>__s | The C string to append.             |
| <br>__n | The number of characters to append. |

**Returns**

Reference to this string.



**append()** [7/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Referenced by `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::basic_string<_CharT>::append()`, `std::__cxx11::collate<_CharT>::do_transform()`, `std::basic_string<_CharT>::operator+=()`, `std::basic_string<_CharT>::operator+=()`, `std::basic_string<_CharT>::operator+=()`, `std::operator>>()`, `std::basic_string<_CharT, st` and `resize()`.

**append()** [8/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**append()** [9/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos) [inline], [constexpr]
```

Append a substring.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | The string to append.                          |
| <code>__pos</code> | Index of the first character of str to append. |
| <code>__n</code>   | The number of characters to append.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                             |
|------------------------------------------------|---------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | if <code>__pos</code> is not a valid index. |
|------------------------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [10/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
 const basic_string<_CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos)
```

Append a substring.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | The string to append.                          |
| <code>__pos</code> | Index of the first character of str to append. |
| <code>__n</code>   | The number of characters to append.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                             |
|------------------------------------------------|---------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | if <code>__pos</code> is not a valid index. |
|------------------------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

**append()** [11/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
 initializer_list<_CharT > __l) [inline], [constexpr]
```

Append an `initializer_list` of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↩ | The initializer_list of characters to append. |
| ↩ |                                               |
| ↩ |                                               |
| ↩ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**append() [12/14]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::append (
 initializer_list<_CharT > __l) [inline]
```

Append an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↩ | The initializer_list of characters to append. |
| ↩ |                                               |
| ↩ |                                               |
| ↩ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**append() [13/14]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::append (
 size_type __n,
 _CharT __c) [inline], [constexpr]
```

Append multiple characters.

**Parameters**

|          |                                     |
|----------|-------------------------------------|
| ↩<br>__n | The number of characters to append. |
| ↩<br>__c | The character to use.               |

**Returns**

Reference to this string.

Appends \_\_n copies of \_\_c to this string.

**append()** [14/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
 size_type __n,
 _CharT __c)
```

Append multiple characters.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__n</code> | The number of characters to append. |
| <code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.

**assign()** [1/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline], [constexpr]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range `[__first,__last)`.

**assign()** [2/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

**assign()** [3/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 basic_string<_CharT, _Traits, _Alloc> && __str) [inline], [constexpr], [noexcept]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [4/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 basic_string<_CharT, _Traits, _Alloc> && __str) [inline], [noexcept]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

**assign()** [5/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 const _CharT * __s) [inline], [constexpr]
```

Set value to contents of a C string.

**Parameters**

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__s</code> | The C string to use. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

**assign() [6/16]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const _CharT * __s) [inline]
```

Set value to contents of a C string.

**Parameters**

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__s</code> | The C string to use. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

**assign() [7/16]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const _CharT * __s,
 size_type __n) [inline], [constexpr]
```

Set value to a C substring.

**Parameters**

|                                                                                                                         |                              |
|-------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__s</code> | The C string to use.         |
| <a href="#"></a><br><code>__n</code> | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

**assign()** [8/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const _CharT * __s,
 size_type __n)
```

Set value to a C substring.

**Parameters**

|                                                                                          |                              |
|------------------------------------------------------------------------------------------|------------------------------|
| <br>__s | The C string to use.         |
| <br>__n | Number of characters to use. |

**Returns**

Reference to this string.

This function sets the value of this string to the first \_\_n characters of \_\_s. If \_\_n is larger than the number of available characters in \_\_s, the remainder of \_\_s is used.

**assign()** [9/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Set value to contents of another string.

**Parameters**

|       |                       |
|-------|-----------------------|
| __str | Source string to use. |
|-------|-----------------------|

**Returns**

Reference to this string.

Referenced by [std::basic\\_string<\\_CharT >::assign\(\)](#), [std::basic\\_string<\\_CharT >::assign\(\)](#), [std::basic\\_string<\\_CharT >::assign\(\)](#), [std::basic\\_string<\\_CharT >::operator=\(\)](#), [std::basic\\_string<\\_CharT >::operator=\(\)](#), [std::basic\\_string<\\_CharT >::operator=\(\)](#), [std::basic\\_string<\\_CharT >::operator=\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf<char >::str\(\)](#) and [std::basic\\_stringbuf<char >::str\(\)](#).

**assign()** [10/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Set value to contents of another string.

**Parameters**

|       |                       |
|-------|-----------------------|
| __str | Source string to use. |
|-------|-----------------------|

**Returns**

Reference to this string.

**assign()** [11/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos) [inline], [constexpr]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [12/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos) [inline]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |



**Returns**

Reference to this string.

**Exceptions**

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | if <i>pos</i> is not a valid index. |
|--------------------------------|-------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

**assign()** [13/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 initializer_list<_CharT> __l) [inline], [constexpr]
```

Set value to an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↵ | The initializer_list of characters to assign. |
| ↵ |                                               |
| ↵ |                                               |
| ↵ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**assign()** [14/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
 initializer_list<_CharT> __l) [inline]
```

Set value to an initializer\_list of characters.

**Parameters**

|   |                                               |
|---|-----------------------------------------------|
| ↵ | The initializer_list of characters to assign. |
| ↵ |                                               |
| ↵ |                                               |
| ↵ |                                               |
| / |                                               |

**Returns**

Reference to this string.

**assign()** [15/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::assign (
```

```
 size_type __n,
 _CharT __c) [inline], [constexpr]
```

Set value to multiple characters.

#### Parameters

|                                                                                                                       |                                 |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__n</code> | Length of the resulting string. |
| <a href="#"></a><br><code>__c</code> | The character to use.           |

#### Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

#### **assign()** [16/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 size_type __n,
 _CharT __c) [inline]
```

Set value to multiple characters.

#### Parameters

|                                                                                                                         |                                 |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__n</code> | Length of the resulting string. |
| <a href="#"></a><br><code>__c</code> | The character to use.           |

#### Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

#### **at()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
 size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

#### Parameters

|                                                                                                                         |                                       |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| <a href="#"></a><br><code>__n</code> | The index of the character to access. |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::at (
 size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

**at()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
 size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__n</code> | The index of the character to access. |
|------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**at()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                                    |                                       |
|------------------------------------|---------------------------------------|
| <code>_↔</code><br><code>_n</code> | The index of the character to access. |
|------------------------------------|---------------------------------------|

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If $n$ is an invalid index. |
|------------------------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

**back()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back () [inline]
```

Returns a read/write reference to the data at the last element of the string.

**back()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

**back()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

**back()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::back () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

**begin()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::begin () [inline]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

**begin()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::begin () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**begin()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**begin()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::begin () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points to the first character in the string.

Referenced by `std::basic_string<_CharT>::crend()`, `std::basic_string<_CharT>::erase()`, `std::basic_string<_CharT>::erase()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::basic_string<_CharT>::insert()`, `std::regex_match()`, `std::regex_match()`, `std::regex_replace()`, `std::regex_replace()`, `std::regex_search()`, `std::basic_string<_CharT>::rend()`, `std::basic_string<_CharT>::rend()`, `std::basic_string<_CharT>::replace()`, `std::basic_string<_CharT>::replace()`, and `std::basic_string<_CharT>::replace()`.

**c\_str()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string<_CharT, _Traits, _Alloc>::c_str () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by `std::__cxx11::collate<_CharT>::do_compare()`, `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::__cxx11::collate<_CharT>::do_transform()`, `std::basic_filebuf<char_type, traits_type>::open()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, `std::operator+()`, and `std::regex_traits<_CharT>::transform_prime()`.

**c\_str()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string<_CharT, _Traits, _Alloc>::c_str () const [inline], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

**capacity()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by [std::basic\\_string<\\_CharT>::assign\(\)](#), [std::basic\\_string<\\_CharT>::push\\_back\(\)](#), [reserve\(\)](#), and [reserve\(\)](#).

**capacity()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

**cbegin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

**cend()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**cend()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**clear()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline], [constexpr], [noexcept]
```

Erases the string, making it empty.

Referenced by [std::\\_\\_cxx11::collate<\\_CharT>::do\\_transform\(\)](#), and [std::basic\\_string<\\_CharT>::operator=\(\)](#).

**clear()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline], [noexcept]
```

Erases the string, making it empty.

**compare()** [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 const _CharT * __s) const [inline], [nodiscard], [constexpr], [noexcept]
```

Compare to a C string.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [2/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 const _CharT * __s) const [inline], [noexcept]
```

Compare to a C string.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [3/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 const basic_string<_CharT, _Traits, _Alloc > & __str) const [inline], [nodiscard],
[constexpr]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [4/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 const basic_string<_CharT, _Traits, _Alloc> & __str) const [inline]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [5/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos,
 size_type __n,
 const basic_string<_CharT, _Traits, _Alloc> & __str) const [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.



**compare()** [6/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 size_type __pos,
 size_type __n,
 const basic_string<_CharT, _Traits, _Alloc > & __str) const [inline]
```

Compare substring to a string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [7/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [8/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const [inline]
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

**compare()** [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

**Parameters**

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare

as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

#### compare() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const [inline]
```

Compare substring against a character array.

#### Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__pos</code> | Index of first character of substring.   |
| <code>__n1</code>  | Number of characters in substring.       |
| <code>__s</code>   | character array to compare against.      |
| <code>__n2</code>  | Number of characters of <code>s</code> . |

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer  $< 0$  if this substring is ordered before the string from `__s`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

#### compare() [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc >::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Alloc > & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__pos1</code> | Index of first character of substring. |
| <code>__n1</code>   | Number of characters in substring.     |
| <code>__str</code>  | String to compare against.             |

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

#### `compare()` [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inline]
```

Compare substring to a substring.

#### Parameters

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

#### Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

#### `copy()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const [constexpr]
```

Copy substring into C string.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

#### Returns

Number of characters actually copied

#### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

#### `copy()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const
```

Copy substring into C string.

#### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

#### Returns

Number of characters actually copied

#### Exceptions

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

**crbegin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crbegin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**crend()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**crend()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**data()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

Referenced by [std::basic\\_regex< char >::assign\(\)](#), [std::basic\\_string< \\_CharT >::compare\(\)](#), [std::basic\\_string< \\_CharT >::compare\(\)](#), [std::basic\\_string< \\_CharT >::compare\(\)](#), [std::\\_\\_cxx11::collate< \\_CharT >::do\\_compare\(\)](#), [std::\\_\\_cxx11::collate< \\_CharT >::do\\_transform\(\)](#), [std::basic\\_string< \\_CharT >::find\(\)](#), [std::basic\\_string< \\_CharT >::find\\_first\\_not\\_of\(\)](#), [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#), [std::basic\\_string< \\_CharT >::find\\_last\\_of\(\)](#), [std::match\\_results< \\_BidirectionalIterator, polymorphic\\_allocator< submatch< \\_BidirectionalIterator > > >::format\(\)](#), [std::wstring\\_convert< \\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc >::from\\_bytes\(\)](#), [std::operator==\(\)](#), [std::basic\\_stringbuf< char >::str\(\)](#), [std::wstring\\_convert< \\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc >::to\\_bytes\(\)](#), and [std::regex\\_traits< \\_CharT >::transform\(\)](#).

**data()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const [inline], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

**data()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () [inline], [nodiscard], [constexpr],
[noexcept]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

**data()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string<_CharT, _Traits, _Alloc>::data () [inline]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

The standard requires this function to be `noexcept` but for the Copy-On-Write string implementation it can throw. This function allows modifying the string contents directly, which means we must copy-on-write to unshare it, which requires allocating memory.

**empty()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string<_CharT, _Traits, _Alloc>::empty () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Referenced by `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, `std::regex_traits<_Ch_type>::lookup_collatename()`, `std::tr2::operator>>()`, `std::basic_string<_CharT>::pop_back()`, and `std::basic_string<_CharT>::pop_back()`.

**empty()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string<_CharT, _Traits, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

**end()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::end () [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

**end()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**end()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

**end()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points one past the last character in the string.

Referenced by [std::basic\\_string< \\_CharT >::append\(\)](#), [std::basic\\_string< \\_CharT >::crbegin\(\)](#), [std::basic\\_string< \\_CharT >::erase\(\)](#), [std::basic\\_string< \\_CharT >::erase\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::rbegin\(\)](#), [std::basic\\_string< \\_CharT >::rbegin\(\)](#), [std::basic\\_string< \\_CharT >::rbegin\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_match\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_replace\(\)](#), [std::regex\\_search\(\)](#), [std::basic\\_string< \\_CharT >::replace\(\)](#), [std::basic\\_string< \\_CharT >::replace\(\)](#), and [std::basic\\_string< \\_CharT >::](#)

**erase()** [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
 __const_iterator __first,
 __const_iterator __last) [inline], [constexpr]
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

**erase()** [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
 __const_iterator __position) [inline], [constexpr]
```

Remove one character.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

**erase()** [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc
>::erase (
```



```
iterator __first,
iterator __last)
```

Remove a range of characters.

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

#### `erase()` [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::erase (
 iterator __position) [inline]
```

Remove one character.

#### Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

#### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

#### `erase()` [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::erase (
 size_type __pos = 0,
 size_type __n = npos) [inline], [constexpr]
```

Remove characters.

#### Parameters

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos</i> is beyond the end of this string. |
|------------------------------------------------|-------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::getline\(\)](#), [std::operator>>\(\)](#), [std::basic\\_string<\\_CharT>::pop\\_back\(\)](#), and [resize\(\)](#).

**erase()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::erase (
 size_type __pos = 0,
 size_type __n = npos) [inline]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos</i> is beyond the end of this string. |
|------------------------------------------------|-------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**find()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to locate. |
|------------------|----------------------|

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Index of character to search from (default 0). |
|--------------------|------------------------------------------------|

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::data\(\)](#), [npos](#), and [std::size\(\)](#).

Referenced by [std::basic\\_string<\\_CharT>::find\(\)](#), [std::basic\\_string<\\_CharT>::find\(\)](#), [std::basic\\_string<\\_CharT>::find\(\)](#), and [std::basic\\_string<\\_CharT>::find\\_first\\_of\(\)](#).

**find()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**find()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__str</code> | String to locate. |
|--------------------|-------------------|

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Index of character to search from (default 0). |
|--------------------|------------------------------------------------|

#### Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### **find\_first\_not\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### **find\_first\_not\_of()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a different character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_first\_not\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character not in C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_not\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
```

```
const _CharT * __s,
size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### **find\_first\_not\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in C string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### **find\_first\_not\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_not_of (
 const basic_string<_CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by `std::basic_string<_CharT>::find_first_not_of()`, `std::basic_string<_CharT>::find_first_not_of()`, and `std::basic_string<_CharT>::find_first_not_of()`.

**find\_first\_not\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

**find\_first\_of()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
```

```
 _CharT __c,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

#### `find_first_of()` [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

#### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

#### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### `find_first_of()` [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_first\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#), and [std::basic\\_string< \\_CharT >::find\\_first\\_of\(\)](#).

**find\_first\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
```

```

 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]

```

Find last position of a different character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### `find_last_not_of()` [2/8]

```

template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [noexcept]

```

Find last position of a different character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_last_not_of()` [3/8]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type basic_string<_CharT, _Traits, _Alloc>::
find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]

```

Find last position of a character not in C substring.

#### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__s</code> | C string containing characters to avoid. |
|------------------|------------------------------------------|

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

#### **find\_last\_not\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character not in C substring.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### **find\_last\_not\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_not\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by `std::basic_string<_CharT>::find_last_not_of()`, `std::basic_string<_CharT>::find_last_not_of()`, and `std::basic_string<_CharT>::find_last_not_of()`.

**find\_last\_not\_of()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_not_of (
```

```
const basic_string<_CharT, _Traits, _Alloc> & __str,
size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `find_last_of()` [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

#### `find_last_of()` [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |



**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

**find\_last\_of()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**find\_last\_of()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**find\_last\_of()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
```

```
const basic_string<_CharT, _Traits, _Alloc> & __str,
size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Referenced by `std::basic_string<_CharT>::find_last_of()`, and `std::basic_string<_CharT>::find_last_of()`.

#### `find_last_of()` [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

#### `front()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::front () [inline]
```

Returns a read/write reference to the data at the first element of the string.

#### `front()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::front () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

#### `front()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

**front()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the first element of the string.

**get\_allocator()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

Referenced by [std::basic\\_string< \\_CharT >::~~basic\\_string\(\)](#), [std::basic\\_string< \\_CharT >::assign\(\)](#), [std::basic\\_string< \\_CharT >::clear\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [std::operator+\(\)](#), [reserve\(\)](#), [reserve\(\)](#), [std::basic\\_string< \\_CharT, std::allocator< \\_CharT > >::swap\(\)](#), and [swap\(\)](#).

**get\_allocator()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

**insert()** [1/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 __const_iterator __p,
 _CharT __c) [inline], [constexpr]
```

Insert one character.

**Parameters**

|                                                                                                                      |                                                       |
|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <a href="#"> <code>__p</code></a> | Iterator referencing position in string to insert at. |
| <a href="#"> <code>__c</code></a> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                   |                                                 |
|-----------------------------------|-------------------------------------------------|
| <a href="#">std::length_error</a> | If new length exceeds <code>max_size()</code> . |
|-----------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [2/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
 const_iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline], [constexpr]
```

Insert a range of characters.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [3/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (
 const_iterator __p,
 initializer_list<_CharT> __l) [inline], [constexpr]
```

Insert an initializer\_list of characters.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__l</code> | The initializer_list of characters to insert.         |

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

**insert()** [4/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 const_iterator __p,
 size_type __n,
 _CharT __c) [inline], [constexpr]
```

Insert multiple characters.

**Parameters**

|                                                                                                                    |                                                             |
|--------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| <a href="#"> <code>__p</code></a> | Const_iterator referencing location in string to insert at. |
| <a href="#"> <code>__n</code></a> | Number of characters to insert                              |
| <a href="#"> <code>__c</code></a> | The character to insert.                                    |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                   |                                                 |
|-----------------------------------|-------------------------------------------------|
| <a href="#">std::length_error</a> | If new length exceeds <code>max_size()</code> . |
|-----------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Referenced by [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), [std::basic\\_string< \\_CharT >::insert\(\)](#), and [std::basic\\_string< \\_CharT >::insert\(\)](#).

**insert()** [5/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 _CharT __c) [inline]
```

Insert one character.

**Parameters**

|                                                                                                                      |                                                       |
|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <a href="#"> <code>__p</code></a> | Iterator referencing position in string to insert at. |
| <a href="#"> <code>__c</code></a> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [6/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
 iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline]
```

Insert a range of characters.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [7/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
 iterator __p,
 initializer_list<_CharT> __l) [inline]
```

Insert an `initializer_list` of characters.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
|------------------|-------------------------------------------------------|

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__i</code> | The initializer_list of characters to insert. |
|------------------|-----------------------------------------------|

### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

### insert() [8/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

### Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### insert() [9/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 const _CharT * __s) [inline], [constexpr]
```

Insert a C string.

### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
|--------------------|----------------------------------|



|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to insert. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                       |
|------------------------------------------------|-------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .       |
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos</code> is beyond the end of this string. |

Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [10/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos,
 const _CharT * __s) [inline]
```

Insert a C string.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
| <code>__s</code>   | The C string to insert.          |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                       |
|------------------------------------------------|-------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .       |
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos</code> is beyond the end of this string. |

Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [11/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n) [inline], [constexpr]
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [12/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n)
```

Insert a C substring.

**Parameters**

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

### `insert()` [13/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline], [constexpr]
```

Insert multiple characters.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

### `insert()` [14/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                         |
|------------------------------------------------|---------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .         |
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [15/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos1,
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Insert value of a string.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [16/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos1,
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Insert value of a string.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [17/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos2,
 size_type __n = npos) [inline], [constexpr]
```

Insert a substring.

**Parameters**

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <code>__pos1</code> | Position in string to insert at.                   |
| <code>__str</code>  | The string to insert.                              |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert. |
| <code>__n</code>    | Number of characters to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                                           |
|------------------------------------------------|---------------------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                           |
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**insert()** [18/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos2,
 size_type __n = npos) [inline]
```

Insert a substring.

**Parameters**

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <code>__pos1</code> | Position in string to insert at.                   |
| <code>__str</code>  | The string to insert.                              |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert. |
| <code>__n</code>    | Number of characters to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                                           |
|------------------------------------------------|---------------------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                           |
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

**length() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length() const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by [std::basic\\_string<\\_CharT>::basic\\_string\(\)](#), [std::basic\\_string<\\_CharT>::basic\\_string\(\)](#), [std::\\_\\_cxx11::collate<\\_CharT>::do\\_transform\(\)](#), [std::basic\\_string<\\_CharT>::operator=\(\)](#), [std::regex\\_replace\(\)](#), and [reserve\(\)](#).

**length() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length() const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

**max\_size() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size() const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the `size()` of the largest possible string.

Referenced by [std::getline\(\)](#), and [std::operator>>\(\)](#).

**max\_size() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size() const [inline], [noexcept]
```

Returns the `size()` of the largest possible string.

**operator+=()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 _CharT __c) [inline], [constexpr]
```

Append a character.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

**Returns**

Reference to this string.

**operator+=()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 _CharT __c) [inline]
```

Append a character.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

**Returns**

Reference to this string.

**operator+=()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 const _CharT * __s) [inline], [constexpr]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**operator+=()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

**operator+=()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator+=()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

**operator+=()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
 initializer_list< _CharT > __l) [inline], [constexpr]
```



Append an initializer\_list of characters.

#### Parameters

|   |                                                    |
|---|----------------------------------------------------|
| ↩ | The initializer_list of characters to be appended. |
| ↩ |                                                    |
| ↩ |                                                    |
| ↩ |                                                    |
| / |                                                    |

#### Returns

Reference to this string.

#### `operator+=()` [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
 initializer_list<_CharT> __l) [inline]
```

Append an initializer\_list of characters.

#### Parameters

|   |                                                    |
|---|----------------------------------------------------|
| ↩ | The initializer_list of characters to be appended. |
| ↩ |                                                    |
| ↩ |                                                    |
| ↩ |                                                    |
| / |                                                    |

#### Returns

Reference to this string.

#### `operator=()` [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 _CharT __c) [inline], [constexpr]
```

Set value to string of length 1.

#### Parameters

|    |                   |
|----|-------------------|
| ↩  | Source character. |
| _C |                   |

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

#### `operator=()` [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 _CharT __c) [inline]
```

Set value to string of length 1.

#### Parameters

|                      |                   |
|----------------------|-------------------|
| $\_ \leftrightarrow$ | Source character. |
| $\_c$                |                   |

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

#### **operator=()** [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move assign the value of *str* to this string.

#### Parameters

|           |                |
|-----------|----------------|
| $\_\_str$ | Source string. |
|-----------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

#### **operator=()** [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [constexpr], [noexcept]
```

Move assign the value of *str* to this string.

#### Parameters

|           |                |
|-----------|----------------|
| $\_\_str$ | Source string. |
|-----------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

#### **operator=()** [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 const _CharT * __s) [inline], [constexpr]
```

Copy contents of *s* into this string.

#### Parameters

|                      |                                |
|----------------------|--------------------------------|
| $\_ \leftrightarrow$ | Source null-terminated string. |
| $\_s$                |                                |

#### **operator=()** [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 const _CharT * __s) [inline]
```

Copy contents of *s* into this string.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Source null-terminated string. |
| <code>_s</code> |                                |

**operator=()** [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline], [constexpr]
```

Assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**operator=()** [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Assign the value of *str* to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

**operator=()** [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 initializer_list<_CharT> __l) [inline], [constexpr]
```

Set value to string constructed from initializer list.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>↔</code>  | std::initializer_list. |
| <code>_↔</code> |                        |
| <code>↔</code>  |                        |
| <code>_↔</code> |                        |
| <code>/</code>  |                        |

**operator=()** [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::operator= (
 initializer_list<_CharT> __l) [inline]
```

Set value to string constructed from initializer list.

**Parameters**

|   |                        |
|---|------------------------|
| ↔ | std::initializer_list. |
| ↔ |                        |
| ↔ |                        |
| ↔ |                        |
| / |                        |

**operator[]() [1/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
 size_type __pos) [inline], [nodiscard], [constexpr]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**operator[]() [2/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
 size_type __pos) [inline]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

**operator[]() [3/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
 size_type __pos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Referenced by `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::back()`, `std::basic_string<_CharT>::front()`, `std::basic_string<_CharT>::front()`, and `std::basic_string<_CharT>::front()`.

**operator[]() [4/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (
 size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**pop\_back() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

**pop\_back() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::pop_back () [inline], [constexpr], [noexcept]
```

Remove the last character.

The string must be non-empty.

**push\_back() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::push_back (
 _CharT __c) [inline], [constexpr]
```

Append a single character.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| <code>_↵</code> | Character to append. |
| <code>_C</code> |                      |

Referenced by [std::\\_\\_cxx11::collate<\\_CharT>::do\\_transform\(\)](#), [std::basic\\_string<\\_CharT>::operator+=\(\(\)\)](#), [std::tr2::operator>>\(\)](#), and [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#).

**push\_back()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::push_back (
 _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| <code>_↵</code> | Character to append. |
| <code>_C</code> |                      |

**rbegin()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

**rbegin()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rbegin()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

**rend()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

**rend()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () const [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**rend()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**rend()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

**replace()** [1/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 const _CharT * __s) [inline], [constexpr]
```

Replace range of characters with C string.

**Parameters**

|                                                                                                       |                                                 |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|
|  <code>__i1</code> | Iterator referencing start of range to replace. |
|  <code>__i2</code> | Iterator referencing end of range to replace.   |
|  <code>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [2/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 const _CharT * __s,
 size_type __n) [inline], [constexpr]
```

Replace range of characters with C substring.

**Parameters**

|                                   |                                                     |
|-----------------------------------|-----------------------------------------------------|
| <a href="#"><code>__i1</code></a> | Iterator referencing start of range to replace.     |
| <a href="#"><code>__i2</code></a> | Iterator referencing end of range to replace.       |
| <a href="#"><code>__s</code></a>  | C string value to insert.                           |
| <a href="#"><code>__n</code></a>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [3/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline], [constexpr]
```

Replace range of characters with string.



**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [4/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 __const_iterator __i1,
 __const_iterator __i2,
 size_type __n,
 _CharT __c) [inline], [constexpr]
```

Replace range of characters with multiple characters.

**Parameters**

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↔<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↔<br/>__i2</code> | Iterator referencing end of range to replace.   |
| <code>↔<br/>__n</code>  | Number of characters to insert.                 |
| <code>↔<br/>__c</code>  | Character to insert.                            |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [5/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 const_iterator __i1,
 const_iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline], [constexpr]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [6/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 const_iterator __i1,
 const_iterator __i2,
 initializer_list< _CharT > __l) [inline], [constexpr]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                         |                                                 |
|-------------------------|-------------------------------------------------|
| <code>↔<br/>__i1</code> | Iterator referencing start of range to replace. |
| <code>↔<br/>__i2</code> | Iterator referencing end of range to replace.   |

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__i</code> | The initializer_list of characters to insert. |
|------------------|-----------------------------------------------|

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [7/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [8/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s) [inline]
```

Replace range of characters with C string.

**Parameters**

|                                                                                                                     |                                                 |
|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"> <code>__i1</code></a> | Iterator referencing start of range to replace. |
| <a href="#"> <code>__i2</code></a> | Iterator referencing end of range to replace.   |
| <a href="#"> <code>__s</code></a>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [9/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s,
 size_type __n) [inline]
```

Replace range of characters with C substring.

**Parameters**

|                                                                                                                       |                                                     |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| <a href="#"> <code>__i1</code></a> | Iterator referencing start of range to replace.     |
| <a href="#"> <code>__i2</code></a> | Iterator referencing end of range to replace.       |
| <a href="#"> <code>__s</code></a>  | C string value to insert.                           |
| <a href="#"> <code>__n</code></a>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [10/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Replace range of characters with string.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [11/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 initializer_list<_CharT> __l) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                                                                                                                        |                                                 |
|------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"></a><br><code>__i1</code> | Iterator referencing start of range to replace. |
| <a href="#"></a><br><code>__i2</code> | Iterator referencing end of range to replace.   |
| <a href="#"></a><br><code>__l</code>  | The initializer_list of characters to insert.   |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace() [12/22]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 size_type __n,
 _CharT __c) [inline]
```

Replace range of characters with multiple characters.

**Parameters**

|                                                                                                                          |                                                 |
|--------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"></a><br><code>__i1</code> | Iterator referencing start of range to replace. |
| <a href="#"></a><br><code>__i2</code> | Iterator referencing end of range to replace.   |
| <a href="#"></a><br><code>__n</code>  | Number of characters to insert.                 |
| <a href="#"></a><br><code>__c</code>  | Character to insert.                            |



```

size_type __n,
const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]

```

Replace characters with value from another string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                                       |
|------------------------------------------------|-------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos</code> is beyond the end of this string. |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### `replace()` [15/22]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline], [constexpr]

```

Replace characters with value of a C string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>pos &gt; size()</code> .               |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |



Removes the characters in the range [`__pos`, `__pos + __n1`) from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### replace() [16/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline]
```

Replace characters with value of a C string.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range [`__pos`, `__pos + __n1`) from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### replace() [17/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::replace
(
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) [inline], [constexpr]
```

Replace characters with value of a C substring.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

|                   |                                            |
|-------------------|--------------------------------------------|
| <code>__n2</code> | Number of characters from <i>s</i> to use. |
|-------------------|--------------------------------------------|

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos1</i> > <code>size()</code> .          |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range [`__pos`, `__pos` + `__n1`) from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [18/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2)
```

Replace characters with value of a C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__pos</code> | Index of first character to replace.       |
| <code>__n1</code>  | Number of characters to be replaced.       |
| <code>__s</code>   | C string to insert.                        |
| <code>__n2</code>  | Number of characters from <i>s</i> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos1</i> > <code>size()</code> .          |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range [`__pos`, `__pos` + `__n1`) from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [19/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc >::replace (
```

```

size_type __pos,
size_type __n1,
size_type __n2,
_CharT __c) [inline], [constexpr]

```

Replace characters with multiple characters.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

#### Returns

Reference to this string.

#### Exceptions

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> .             |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

#### `replace()` [20/22]

```

template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos,
 size_type __n1,
 size_type __n2,
 _CharT __c) [inline]

```

Replace characters with multiple characters.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                 |
|------------------------------------------------|-------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> .             |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [21/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Alloc> & __str,
 size_type __pos2,
 size_type __n2 = npos) [inline], [constexpr]
```

Replace characters with value from another string.

**Parameters**

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <code>__pos1</code> | Index of first character to replace.                 |
| <code>__n1</code>   | Number of characters to be replaced.                 |
| <code>__str</code>  | String to insert.                                    |
| <code>__pos2</code> | Index of first character of <code>str</code> to use. |
| <code>__n2</code>   | Number of characters from <code>str</code> to use.   |

**Returns**

Reference to this string.

**Exceptions**

|                                                |                                                                               |
|------------------------------------------------|-------------------------------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

**replace()** [22/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos1,
 size_type __n1,
```

```
const basic_string< _CharT, _Traits, _Alloc > & __str,
size_type __pos2,
size_type __n2 = npos) [inline]
```

Replace characters with value from another string.

### Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

### Returns

Reference to this string.

### Exceptions

|                                                |                                                                               |
|------------------------------------------------|-------------------------------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <a href="#"><code>std::length_error</code></a> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

### `reserve()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve () [constexpr]
```

Equivalent to `shrink_to_fit()`.

References [`capacity\(\)`](#), [`get\_allocator\(\)`](#), and [`length\(\)`](#).

Referenced by [`std::basic\_string< \_CharT >::push\_back\(\)`](#), and [`std::basic\_string< \_CharT >::shrink\_to\_fit\(\)`](#).

### `reserve()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()
```

Equivalent to `shrink_to_fit()`.

### `reserve()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
 size_type __res_arg) [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

### Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                                |                                                             |
|------------------------------------------------|-------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

References [`capacity\(\)`](#), [`get\_allocator\(\)`](#), and [`std::size\(\)`](#).

Referenced by [`std::num\_get<\_CharT, \_InIter>::do\_get\(\)`](#), [`std::num\_get<\_CharT, \_InIter>::do\_get\(\)`](#), [`std::num\_get<\_CharT, \_InIter>::do\_get\(\)`](#), [`std::tr2::operator>>\(\)`](#), and [`std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::overflow\(\)`](#).

**reserve()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::reserve (
 size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                                |                                                             |
|------------------------------------------------|-------------------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|------------------------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

**resize()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
 size_type __n) [inline], [constexpr]
```

Resizes the string to the specified number of characters.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

**resize()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
 size_type __n) [inline]
```

Resizes the string to the specified number of characters.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| $\leftrightarrow$<br>__n | Number of characters the string should contain. |
|--------------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

**resize()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
 size_type __n,
 _CharT __c) [constexpr]
```

Resizes the string to the specified number of characters.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| $\leftrightarrow$<br>__n | Number of characters the string should contain. |
| $\leftrightarrow$<br>__c | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to \_\_c.

References [append\(\)](#), [erase\(\)](#), and [size\(\)](#).

Referenced by [std::basic\\_string< \\_CharT >::resize\(\)](#).

**resize()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
 size_type __n,
 _CharT __c)
```

Resizes the string to the specified number of characters.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| $\leftrightarrow$<br>__n | Number of characters the string should contain. |
| $\leftrightarrow$<br>__c | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to \_\_c.

**rfind()** [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
 _CharT __c,
 size_type __pos = npos) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

References [npos](#), and [std::size\(\)](#).

**rfind()** [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

**rfind()** [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.



**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

References [std::min\(\)](#), [npos](#), and [std::size\(\)](#).

**rfind()** [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

**rfind()** [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | C string to locate. |
|------------------|---------------------|

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__pos</code> | Index of character to start search at (default end). |
|--------------------|------------------------------------------------------|

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `rfind()` [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a C string.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

#### `rfind()` [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

#### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by `std::basic_string< _CharT >::find_last_of()`, `std::basic_string< _CharT >::rfind()`, `std::basic_string< _CharT >::rfind()`, `std::basic_string< _CharT >::rfind()`, and `std::basic_string< _CharT >::rfind()`.

**rfind()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

### Find last position of a string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

## shrink\_to\_fit() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [constexpr], [noexcept]
A non-binding request to reduce capacity() to size().
```

## shrink\_to\_fit() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [noexcept]
A non-binding request to reduce capacity() to size().
```

```
size() [1/2]
```

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::dynamic\_bitset(), std::basic\_string< \_CharT >::append(),  
std::basic\_string< \_CharT >::append(), std::basic\_regex< char >::assign(), std::basic\_string< \_CharT >::assign(),  
std::basic\_string< \_CharT >::assign(), std::basic\_string< \_CharT >::assign(), std::basic\_string< \_CharT >::assign(),  
std::basic\_string< \_CharT >::assign(), std::basic\_string< \_CharT >::assign(), std::basic\_string< \_CharT >::at(),  
std::basic\_string< \_CharT >::at(), std::basic\_string< \_CharT >::back(), std::basic\_string< \_CharT >::back(),  
std::basic\_string< \_CharT >::back(), std::basic\_string< \_CharT >::cend(), std::basic\_string< \_CharT >::compare(),  
std::basic\_string< \_CharT >::compare(), std::basic\_string< \_CharT >::compare(), std::basic\_string< \_CharT >::empty(),  
std::basic\_string< \_CharT >::end(), std::basic\_string< \_CharT >::end(), std::basic\_string< \_CharT >::end(),  
std::basic\_string< \_CharT >::find(), std::basic\_string< \_CharT >::find\_first\_not\_of(), std::basic\_string< \_CharT >::find\_first\_of(),  
std::basic\_string< \_CharT >::find\_last\_of(), std::match\_results< \_BidirectionalIterator, polymorphic\_allocator< sub-  
\_match< \_BidirectionalIterator > >>>::format(), std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::fr  
std::basic\_string< \_CharT >::insert(), std::basic\_string< \_CharT >::length(), std::operator+(), std::operator+(),  
std::operator+(), std::operator+(), std::operator+(), std::basic\_string< \_CharT >::operator=(), std::operator==(,  
std::tr2::operator>>(), std::basic\_string< \_CharT >::operator[], std::basic\_string< \_CharT >::operator[](), std::basic\_string< \_CharT  
std::basic\_string< \_CharT >::pop\_back(), std::basic\_string< \_CharT >::push\_back(), std::basic\_string< \_CharT >::replace(),  
std::basic\_string< \_CharT >::replace(), resize(), std::basic\_stringbuf< char >::str(), std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc  
std::regex\_traits< CharT >::transform(), and std::regex\_traits< CharT >::transform\_primary().

**size()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const [inline], [noexcept]
Returns the number of characters in the string, not including any null-termination.
```

**substr()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr]
```

Get a substring.

**Parameters**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**substr()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline]
```

Get a substring.

**Parameters**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                                |                                     |
|------------------------------------------------|-------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> . |
|------------------------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**subview()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view<_CharT, _Traits> std::basic_string<_CharT, _Traits, _Alloc>::subview (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr]
```

Get a subview.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                |
| <code>__n</code>   | Number of characters in subview (default remainder). |

**Returns**

The subview.

**Exceptions**

|                                                |                                     |
|------------------------------------------------|-------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos &gt; size()</code> . |
|------------------------------------------------|-------------------------------------|

Construct and return a subview using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**subview()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view<_CharT, _Traits> std::basic_string<_CharT, _Traits, _Alloc>::subview (
 size_type __pos = 0,
 size_type __n = npos) const [inline], [nodiscard], [constexpr]
```

Get a subview.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                |
| <code>__n</code>   | Number of characters in subview (default remainder). |

**Returns**

The subview.

**Exceptions**

|                                   |                                     |
|-----------------------------------|-------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>__pos &gt; size()</code> . |
|-----------------------------------|-------------------------------------|

Construct and return a subview using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

**swap() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
 basic_string< _CharT, _Traits, _Alloc > & __s) [constexpr], [noexcept]
```

Swap contents with another string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

References [basic\\_string\(\)](#), and [get\\_allocator\(\)](#).

Referenced by [std::basic\\_string< \\_CharT >::assign\(\)](#), and [std::basic\\_string< \\_CharT >::operator=\(\)](#).

**swap() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
 basic_string< _CharT, _Traits, _Alloc > & __s) [noexcept]
```

Swap contents with another string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

**6.249.4 Member Data Documentation****npos**

```
template<typename _CharT, typename _Traits, typename _Alloc>
const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos [static]
```

Value returned by various member functions when they fail.

Referenced by [std::bitset< \\_Nb >::bitset\(\)](#), [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >::dynamic\\_bitset\(\)](#), [find\(\)](#), [find\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_not\\_of\(\)](#), [find\\_first\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_not\\_of\(\)](#), [find\\_last\\_of\(\)](#), [rfind\(\)](#), and [rfind\(\)](#).

The documentation for this class was generated from the following files:

- [basic\\_string.h](#)

- [cow\\_string.h](#)
- [basic\\_string.tcc](#)

## 6.250 `std::basic_string_view< _CharT, _Traits >` Class Template Reference

```
#include <>>
```

### Public Types

- using `const_iterator`
- using `const_pointer`
- using `const_reference`
- using `const_reverse_iterator`
- using `difference_type`
- using `iterator`
- using `pointer`
- using `reference`
- using `reverse_iterator`
- using `size_type`
- using `traits_type`
- using `value_type`

### Public Member Functions

- `template<contiguous_iterator _It, sized_sentinel_for< _It > _End>`  
`constexpr basic_string_view ( _It __first, _End __last) noexcept(noexcept(__last - __first))`
- `template<typename _Range, typename _DRange = remove_cvref_t< _Range >>`  
`constexpr basic_string_view ( _Range &&__r) noexcept(noexcept(ranges::size(__r)) &&noexcept(ranges::data(__r)))`
- `constexpr basic_string_view (const _CharT * __str) noexcept`
- `constexpr basic_string_view (const _CharT * __str, size_type __len) noexcept`
- `constexpr basic_string_view (const basic_string_view &) noexcept=default`
- `basic_string_view (nullptr_t)=delete`
- `constexpr const_reference at (size_type __pos) const`
- `constexpr const_reference back () const noexcept`
- `constexpr const_iterator begin () const noexcept`
- `constexpr const_iterator cbegin () const noexcept`
- `constexpr const_iterator cend () const noexcept`
- `constexpr int compare (basic_string_view __str) const noexcept`
- `constexpr int compare (const _CharT * __str) const noexcept`
- `constexpr int compare (size_type __pos1, size_type __n1, basic_string_view __str) const`
- `constexpr int compare (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const _CharT * __str) const`
- `constexpr int compare (size_type __pos1, size_type __n1, const _CharT * __str, size_type __n2) const noexcept(false)`
- `constexpr bool contains ( _CharT __x) const noexcept`
- `constexpr bool contains (basic_string_view __x) const noexcept`
- `constexpr bool contains (const _CharT * __x) const noexcept`
- `constexpr size_type copy ( _CharT * __str, size_type __n, size_type __pos=0) const`
- `constexpr const_reverse_iterator crbegin () const noexcept`
- `constexpr const_reverse_iterator crend () const noexcept`

- constexpr const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr const\_reference **front** () const noexcept
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr basic\_string\_view & **operator=** (const basic\_string\_view &) noexcept=default
- constexpr const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n) noexcept
- constexpr void **remove\_suffix** (size\_type \_\_n) noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **rfind** (basic\_string\_view \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr basic\_string\_view **substr** (size\_type \_\_pos=0, size\_type \_\_n=npo) const noexcept(false)
- constexpr basic\_string\_view **subview** (size\_type \_\_pos=0, size\_type \_\_n=npo) const
- constexpr void **swap** (basic\_string\_view & \_\_sv) noexcept

## Static Public Attributes

- static constexpr size\_type **npo**



### 6.250.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::basic_string_view<_CharT, _Traits>
```

A non-owning reference to a string.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A `basic_string_view` looks like this:

```
_CharT* _M_str
size_t _M_len
```

The documentation for this class was generated from the following files:

- [string\\_view](#)
- [bits/string\\_view.tcc](#)

## 6.251 std::experimental::filesystem::v1::basic\_string\_view<\_CharT, \_Traits> Class Template Reference

```
#include <>>
```

### Public Types

- using **const\_iterator**
- using **const\_pointer**
- using **const\_reference**
- using **const\_reverse\_iterator**
- using **difference\_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse\_iterator**
- using **size\_type**
- using **traits\_type**
- using **value\_type**

### Public Member Functions

- template<contiguous\_iterator \_It, sized\_sentinel\_for<\_It> \_End>  
constexpr **basic\_string\_view** (\_It \_\_first, \_End \_\_last) noexcept(noexcept(\_\_last - \_\_first))
- template<typename \_Range, typename \_DRange = remove\_cvref\_t<\_Range>>  
constexpr **basic\_string\_view** (\_Range &&\_\_r) noexcept(noexcept(ranges::size(\_\_r)) &&noexcept(ranges::data(\_\_r)))
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str) noexcept
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len) noexcept
- constexpr **basic\_string\_view** (const basic\_string\_view &) noexcept=default
- **basic\_string\_view** (nullptr\_t)=delete
- constexpr const\_reference **at** (size\_type \_\_pos) const
- constexpr const\_reference **back** () const noexcept
- constexpr const\_iterator **begin** () const noexcept

- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr int **compare** (basic\_string\_view \_\_str) const noexcept
- constexpr int **compare** (const \_CharT \* \_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \* \_\_str, size\_type \_\_n2) const noexcept(false)
- constexpr bool **contains** (\_CharT \_\_x) const noexcept
- constexpr bool **contains** (basic\_string\_view \_\_x) const noexcept
- constexpr bool **contains** (const \_CharT \* \_\_x) const noexcept
- constexpr size\_type **copy** (\_CharT \* \_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr const\_reference **front** () const noexcept
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr basic\_string\_view & **operator=** (const basic\_string\_view &) noexcept=default
- constexpr const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n) noexcept
- constexpr void **remove\_suffix** (size\_type \_\_n) noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept

- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr basic\_string\_view **substr** (size\_type \_\_pos=0, size\_type \_\_n=npos) const noexcept(false)
- constexpr basic\_string\_view **subview** (size\_type \_\_pos=0, size\_type \_\_n=npos) const
- constexpr void **swap** (basic\_string\_view &\_\_sv) noexcept

### Static Public Attributes

- static constexpr size\_type **npos**

#### 6.251.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::filesystem::v1::basic_string_view<_CharT, _Traits>
```

A non-owning reference to a string.

### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A `basic_string_view` looks like this:

```
_CharT* _M_str
size_t _M_len
```

The documentation for this class was generated from the following files:

- [string\\_view](#)
- [bits/string\\_view.tcc](#)

## 6.252 `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>` Class Template Reference

```
#include <>>
```

### Public Types

- using **const\_iterator**
- using **const\_pointer**
- using **const\_reference**
- using **const\_reverse\_iterator**
- using **difference\_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse\_iterator**
- using **size\_type**
- using **traits\_type**
- using **value\_type**

## Public Member Functions

- constexpr **basic\_string\_view** (const \_CharT \*\_\_str)
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len)
- template<typename \_Allocator>  
**basic\_string\_view** (const [basic\\_string](#)< \_CharT, \_Traits, \_Allocator > &\_\_str) noexcept
- constexpr **basic\_string\_view** (const basic\_string\_view &) noexcept=default
- constexpr const \_CharT & **at** (size\_type \_\_pos) const
- constexpr const \_CharT & **back** () const
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr int **compare** (basic\_string\_view \_\_str) const noexcept
- constexpr int **compare** (const \_CharT \*\_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, basic\_string\_view \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str, size\_type \_\_n2) const
- size\_type **copy** (\_CharT \*\_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- constexpr const \_CharT \* **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (basic\_string\_view \_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=npow) const noexcept
- constexpr const \_CharT & **front** () const
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- template<typename \_Allocator>  
**operator basic\_string**< \_CharT, \_Traits, \_Allocator > () const
- basic\_string\_view & **operator=** (const basic\_string\_view &) noexcept=default

- constexpr const \_CharT & **operator[]** (size\_type \_\_pos) const
- const\_reverse\_iterator **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n)
- constexpr void **remove\_suffix** (size\_type \_\_n)
- const\_reverse\_iterator **rend** () const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **rfind** (basic\_string\_view \_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \*\_\_str, size\_type \_\_pos=npos) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr basic\_string\_view **substr** (size\_type \_\_pos=0, size\_type \_\_n=npos) const
- constexpr void **swap** (basic\_string\_view &\_\_sv) noexcept
- template<typename \_Allocator = std::allocator<\_CharT>>  
[basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **to\_string** (const \_Allocator &\_\_alloc=\_Allocator()) const

### Static Public Attributes

- static constexpr size\_type **npos**

#### 6.252.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
```

A non-owning reference to a string.

### Template Parameters

|                |                                                             |
|----------------|-------------------------------------------------------------|
| <i>_CharT</i>  | Type of character                                           |
| <i>_Traits</i> | Traits for character type, defaults to char_traits<_CharT>. |

A basic\_string\_view looks like this:

```
_CharT* _M_str
size_t _M_len
```

The documentation for this class was generated from the following files:

- [experimental/string\\_view](#)
- [experimental/bits/string\\_view.tcc](#)

## 6.253 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for `std::basic_stringbuf< _CharT, _Traits, _Alloc >`:



### Public Types

- `typedef __string_type::size_type __size_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef basic_string< char_type, _Traits, _Alloc > __string_type`
- `typedef _Alloc allocator_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef _Traits traits_type`

### Public Member Functions

- `basic_stringbuf ()`
- `basic_stringbuf (__string_type && __s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (basic_stringbuf && __rhs)`
- `basic_stringbuf (basic_stringbuf && __rhs, const allocator_type & __a)`
- `basic_stringbuf (const __string_type & __str, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (const allocator_type & __a)`
- `template<typename _SAlloc>`  
`basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > & __s, const allocator_type & __a)`
- `template<typename _SAlloc>`  
`basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > & __s, ios_base::openmode __mode, const allocator_type & __a)`
- `template<typename _SAlloc>`  
`basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > & __s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (const basic_stringbuf &)=delete`
- `basic_stringbuf (ios_base::openmode __mode)`
- `basic_stringbuf (ios_base::openmode __mode, const allocator_type & __a)`
- `allocator_type get_allocator () const noexcept`
- `locale getloc () const`

- [streamsize in\\_avail \(\)](#)
  - [basic\\_stringbuf & operator= \(basic\\_stringbuf &&\\_\\_rhs\)](#)
  - [basic\\_stringbuf & operator= \(const basic\\_stringbuf &\)=delete](#)
  - [locale pubimbue \(const locale &\\_\\_loc\)](#)
  - [int\\_type sbumpc \(\)](#)
  - [int\\_type sgetc \(\)](#)
  - [streamsize sgetn \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [int\\_type snextc \(\)](#)
  - [int\\_type sputbackc \(char\\_type \\_\\_c\)](#)
  - [int\\_type sputc \(char\\_type \\_\\_c\)](#)
  - [streamsize sputn \(const char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [\\_\\_string\\_type str \(\) &&](#)
  - [\\_\\_string\\_type str \(\) const &](#)
  - [void str \(\\_\\_string\\_type &&\\_\\_s\)](#)
  - [void str \(const \\_\\_string\\_type &\\_\\_s\)](#)
  - [template<\\_\\_allocator\\_like \\_SAlloc>  
basic\\_string< \\_CharT, \\_Traits, \\_SAlloc > str \(const \\_SAlloc &\\_\\_sa\) const](#)
  - [template<\\_\\_allocator\\_like \\_SAlloc>  
void str \(const basic\\_string< \\_CharT, \\_Traits, \\_SAlloc > &\\_\\_s\)](#)
  - [int\\_type sungetc \(\)](#)
  - [void swap \(basic\\_stringbuf &\\_\\_rhs\) noexcept\(\\_Noexcept\\_swap::value\)](#)
  - [basic\\_string\\_view< char\\_type, traits\\_type > view \(\) const noexcept](#)
- 
- [basic\\_streambuf \\* pubsetbuf \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
  - [pos\\_type pubseekoff \(off\\_type \\_\\_off, ios\\_base::seekdir \\_\\_way, ios\\_base::openmode \\_\\_mode=ios\\_base::in|ios\\_base::out\)](#)
  - [pos\\_type pubseekpos \(pos\\_type \\_\\_sp, ios\\_base::openmode \\_\\_mode=ios\\_base::in|ios\\_base::out\)](#)
  - [int pubsync \(\)](#)

### Protected Member Functions

- [void \\_\\_safe\\_gbump \(streamsize \\_\\_n\)](#)
- [void \\_\\_safe\\_pbump \(streamsize \\_\\_n\)](#)
- [void \\_M\\_pbump \(char\\_type \\*\\_\\_pbeg, char\\_type \\*\\_\\_pend, off\\_type \\_\\_off\)](#)
- [void \\_M\\_stringbuf\\_init \(ios\\_base::openmode \\_\\_mode\)](#)
- [void \\_M\\_sync \(char\\_type \\*\\_\\_base, \\_\\_size\\_type \\_\\_i, \\_\\_size\\_type \\_\\_o\)](#)
- [void \\_M\\_update\\_egptr \(\)](#)
- [void gbump \(int \\_\\_n\)](#)
- [virtual void imbue \(const locale &\\_\\_loc\)](#)
- [virtual int\\_type overflow \(int\\_type \\_\\_c=traits\\_type::eof\(\)\)](#)
- [virtual int\\_type pbackfail \(int\\_type \\_\\_c=traits\\_type::eof\(\)\)](#)
- [void pbump \(int \\_\\_n\)](#)
- [virtual pos\\_type seekoff \(off\\_type \\_\\_off, ios\\_base::seekdir \\_\\_way, ios\\_base::openmode \\_\\_mode=ios\\_base::in|ios\\_base::out\)](#)
- [virtual pos\\_type seekpos \(pos\\_type \\_\\_sp, ios\\_base::openmode \\_\\_mode=ios\\_base::in|ios\\_base::out\)](#)
- [virtual \\_\\_streambuf\\_type \\* setbuf \(char\\_type \\*\\_\\_s, streamsize \\_\\_n\)](#)
- [void setg \(char\\_type \\*\\_\\_gbeg, char\\_type \\*\\_\\_gnext, char\\_type \\*\\_\\_gend\)](#)
- [void setp \(char\\_type \\*\\_\\_pbeg, char\\_type \\*\\_\\_pend\)](#)
- [virtual streamsize showmanyc \(\)](#)
- [void swap \(basic\\_streambuf &\\_\\_sb\)](#)
- [virtual int sync \(\)](#)
- [virtual int\\_type uflow \(\)](#)
- [virtual int\\_type underflow \(\)](#)

- virtual [streamsize xsgetn](#) (char\_type \*\_\_s, [streamsize \\_\\_n](#))
- virtual [streamsize xspn](#) (const char\_type \*\_\_s, [streamsize \\_\\_n](#))
- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const
- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

### Protected Attributes

- [locale \\_M\\_buf\\_locale](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- [ios\\_base::openmode \\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)
- \_\_string\_type [\\_M\\_string](#)

### 6.253.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for std::string).

### Template Parameters

|                         |                                                             |
|-------------------------|-------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                   |
| <a href="#">_Traits</a> | Traits for character type, defaults to char_traits<_CharT>. |
| <a href="#">_Alloc</a>  | Allocator type, defaults to allocator<_CharT>.              |

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a std::basic\_string. (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type ios\_base::openmode) have in set if the input sequence can be read, and out set if the output sequence can be written.

### 6.253.2 Constructor & Destructor Documentation

#### basic\_stringbuf() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf () [inline]
```

Starts with an empty string buffer.

The default constructor initializes the parent class using its own default ctor.



**basic\_stringbuf()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

The default constructor initializes the parent class using its own default ctor.

**basic\_stringbuf()** [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

**Parameters**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

This constructor initializes the parent class using its own default ctor.

**6.253.3 Member Function Documentation****eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**egptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >:: M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char_type, traits_type >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

### epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputc()`, `std::basic_filebuf< _CharT, _Traits >::xsputn()`, and `xsputn()`.

### gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

### getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

## Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

## gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr \(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_create\\_pback\(\)](#), [std::basic\\_filebuf< char\\_type, traits\\_type >::\\_M\\_destroy](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::in\\_avail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sgetc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#), [std::basic\\_stringbuf< char >::showmanyc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::uflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::underflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::underflow\(\)](#), [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >::underflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#), and [xsgetn\(\)](#).

## imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue \(
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [std::basic\\_filebuf< wchar\\_t >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::overflow (
 int_type __c = traits_type::eof()) [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>_↵<br/>_c</code> | An additional character to consume. |
|------------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::eptr()`, `std::basic_streambuf< _CharT, _Traits >::in`, `std::max()`, `std::min()`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, `std::basic_streambuf< _CharT, _Traits >::pptr()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_streambuf< _CharT, _Traits >::setg()`.

**pbackfail()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::pbackfail (
 int_type __c = traits_type::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↔</code> | The character to be inserted back into the sequence. |
| <code>_C</code> |                                                      |

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

References [\\_M\\_mode](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::eback\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::gbump\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::gptr\(\)](#), and [std::ios\\_base::out](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::str\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↔</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), and [std::basic\\_streambuf<char\\_type, traits\\_type>::sputc\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_streambuf< char\\_type, \\_Traits >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived [imbue\(\\_\\_loc\)](#).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                                |
|---------------------|------------------------------------------------|
| <code>__off</code>  | Offset.                                        |
| <code>__way</code>  | Value for <a href="#">ios_base::seekdir</a> .  |
| <code>__mode</code> | Value for <a href="#">ios_base::openmode</a> . |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
pos_type __sp,
ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

#### pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

#### pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

#### sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

#### seekoff()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::e`, `std::ios_base::end`, `std::basic_streambuf< _CharT, _Traits >::epptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::ios_base::in`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pptr()`, and `std::basic_streambuf< _CharT, _Traits >::setg()`.

**seekpos()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::epptr()`, `std::ios_base::in`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()` and `std::basic_streambuf< _CharT, _Traits >::setg()`.

**setbuf()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type * std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
 char_type * __s,
 streamsize __n) [inline], [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no buffer has already been created, and both `__s` and `__n` are non-zero, then `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.↵buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
```



```

char_type * __gbeg,
char_type * __gnext,
char_type * __gend) [inline], [protected], [inherited]

```

Setting the three read area pointers.

#### Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

#### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert()`, `std::basic_filebuf< char_type, traits_type >::M_create()`, `std::basic_filebuf< char_type, traits_type >::M_destroy_pback()`, `std::basic_filebuf< char_type, traits_type >::M_set_buffer()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

#### setp()

```

template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]

```

Setting the three write area pointers.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

#### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == ep_ptr()`

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert()`, and `std::basic_filebuf< char_type, traits_type >::M_s`

#### sgetc()

```

template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]

```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

## sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for xsgetn.

### Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

## showmanyc()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc () [inline], [protected],
[virtual]
```

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

## snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

### Returns

The next character, or eof.

Calls `sputc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, char_traits< _CharT > >::get()`, and `std::basic_istream< _CharT, char_traits< _CharT > >::readsome()`.

**sputback()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputback (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::putc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                          |                     |
|--------------------------|---------------------|
| $\leftrightarrow$<br>__s | A buffer read area. |
| $\leftrightarrow$<br>__n | A count.            |

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**str() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str () const & [inline], [nodiscard]
```

Copying out the string buffer.

**Returns**

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1*

**str() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__s | The string to use as a new sequence. |
|--------------------------|--------------------------------------|

Deallocates any previous stored sequence, then copies `s` to use as a new one.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream< _CharT, _Traits >::unget()`.

**sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync () [inline], [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), [std::basic\\_filebuf< wchar\\_t >](#), and [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubsync\(\)](#).

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), and [xsgetn\(\)](#).

**underflow()**

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::underflow () [protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [\\_M\\_mode](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::egptr\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::gptr\(\)](#), and [std::ios\\_base::in](#).

**xsggetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsggetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                                                                                                       |                                         |
|-------------------------------------------------------------------------------------------------------|-----------------------------------------|
| <br><code>__s</code> | A buffer area.                          |
| <br><code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

References [egptr\(\)](#), [gptra\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sggetn\(\)](#).

**xspuotn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xspuotn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                                                                                                         |                                        |
|---------------------------------------------------------------------------------------------------------|----------------------------------------|
| <br><code>__s</code> | A buffer area.                         |
| <br><code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< \\_CharT, std::char\\_traits< \\_CharT > >](#), [std::basic\\_filebuf< char >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

References [epptr\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptr\(\)](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::spuotn\(\)](#).

#### 6.253.4 Member Data Documentation

##### **\_M\_buf\_locale**

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::basic\\_filebuf\(\)](#).

##### **\_M\_in\_beg**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

##### **\_M\_in\_cur**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

##### **\_M\_in\_end**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

##### **\_M\_mode**

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current stringbuf.

Referenced by [overflow\(\)](#), [pbackfail\(\)](#), [seekoff\(\)](#), [seekpos\(\)](#), and [underflow\(\)](#).

##### **\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

##### **\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

##### **\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

The documentation for this class was generated from the following files:

- [sstream](#)
- [sstream.tcc](#)

## 6.254 std::basic\_stringstream<\_CharT, \_Traits, \_Alloc> Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>:



### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_iostream<char_type, traits_type>` `__iostream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`



- typedef [basic\\_stringbuf](#)< \_CharT, \_Traits, \_Alloc > [\\_\\_stringbuf\\_type](#)
- typedef \_Alloc [allocator\\_type](#)
- typedef \_CharT [char\\_type](#)
- enum [event](#) { [erase\\_event](#) , [imbue\\_event](#) , [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_Ios\_Fmtflags [fmtflags](#)
- typedef traits\_type::int\_type [int\\_type](#)
- typedef \_Ios\_Iostate [iostate](#)
- typedef traits\_type::off\_type [off\\_type](#)
- typedef \_Ios\_Openmode [openmode](#)
- typedef traits\_type::pos\_type [pos\\_type](#)
- typedef \_Ios\_Seekdir [seekdir](#)
- typedef \_Traits [traits\\_type](#)

### Public Member Functions

- [basic\\_stringstream](#) ()
- **[basic\\_stringstream](#)** (\_\_string\_type &&\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- **[basic\\_stringstream](#)** ([basic\\_stringstream](#) &&\_\_rhs)
- [basic\\_stringstream](#) (const \_\_string\_type &\_\_str, [ios\\_base::openmode](#) \_\_m=[ios\\_base::out](#)|[ios\\_base::in](#))
- template<typename \_SAlloc>  
**[basic\\_stringstream](#)** (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_str, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
**[basic\\_stringstream](#)** (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_str, [ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- template<typename \_SAlloc>  
**[basic\\_stringstream](#)** (const [basic\\_string](#)< \_CharT, \_Traits, \_SAlloc > &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- **[basic\\_stringstream](#)** (const [basic\\_stringstream](#) &)=delete
- [basic\\_stringstream](#) ([ios\\_base::openmode](#) \_\_m)
- **[basic\\_stringstream](#)** ([ios\\_base::openmode](#) \_\_mode, const allocator\_type &\_\_a)
- [~basic\\_stringstream](#) ()
- template<typename \_ValueT>  
[basic\\_istream](#)< \_CharT, \_Traits > & [\\_M\\_extract](#) (\_ValueT &\_\_v)
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT>  
[basic\\_ostream](#)< \_CharT, \_Traits > & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- \_\_ostream\_type & [flush](#) ()
- [streamsize](#) [gcount](#) () const

- `basic_istream< wchar_t > & getline (char_type * __s, streamsize __n, char_type __delim)`
- `basic_istream< char > & getline (char_type * __s, streamsize __n, char_type __delim)`
- `locale getloc () const`
- `bool good () const`
- `basic_istream< wchar_t > & ignore (streamsize __n)`
- `basic_istream< char > & ignore (streamsize __n)`
- `basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale & __loc)`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `__ostream_type & operator<< (__streambuf_type * __sb)`
- `__ostream_type & operator<< (const void * __p)`
- `__ostream_type & operator<< (const volatile void * __p)`
- `__ostream_type & operator<< (nullptr_t)`
- `basic_stringstream & operator= (basic_stringstream && __rhs)`
- `basic_stringstream & operator= (const basic_stringstream &)=delete`
- `__istream_type & operator>> (__streambuf_type * __sb)`
- `__istream_type & operator>> (void *& __p)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
- `__stringbuf_type * rdbuf () const`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `__ostream_type & seekp (pos_type)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `__string_type str () &&`
- `__string_type str () const &`
- `void str (__string_type && __s)`
- `void str (const __string_type & __s)`
- `template<__allocator_like _SAlloc>`  
`basic_string< _CharT, _Traits, _SAlloc > str (const _SAlloc & __sa) const`
- `template<__allocator_like _SAlloc>`  
`void str (const basic_string< _CharT, _Traits, _SAlloc > & __s)`
- `void swap (basic_stringstream & __rhs)`
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie () const`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
- `void unsetf (fmtflags __mask)`
- `basic_string_view< char_type, traits_type > view () const noexcept`
- `char_type widen (char __c) const`
- `streamsize width () const`
- `streamsize width (streamsize __wide)`
- `__istream_type & operator>> (__istream_type & (* __pf) (__istream_type &))`

- \_\_istream\_type & [operator>>](#) (\_\_ios\_type &(\_\_pf)(\_\_ios\_type &))
- \_\_istream\_type & [operator>>](#) (ios\_base &(\_\_pf)(ios\_base &))

### Extractors

All the *operator>>* functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- \_\_istream\_type & [operator>>](#) (bool &\_\_n)
  - \_\_istream\_type & [operator>>](#) (short &\_\_n)
  - \_\_istream\_type & [operator>>](#) (unsigned short &\_\_n)
  - \_\_istream\_type & [operator>>](#) (int &\_\_n)
  - \_\_istream\_type & [operator>>](#) (unsigned int &\_\_n)
  - \_\_istream\_type & [operator>>](#) (long &\_\_n)
  - \_\_istream\_type & [operator>>](#) (unsigned long &\_\_n)
  - \_\_istream\_type & [operator>>](#) (long long &\_\_n)
  - \_\_istream\_type & [operator>>](#) (unsigned long long &\_\_n)
- 
- \_\_istream\_type & [operator>>](#) (float &\_\_f)
  - \_\_istream\_type & [operator>>](#) (double &\_\_f)
  - \_\_istream\_type & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- int\_type [get](#) ()
- \_\_istream\_type & [get](#) (char\_type &\_\_c)
- \_\_istream\_type & [get](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- \_\_istream\_type & [get](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- \_\_istream\_type & [get](#) (\_\_streambuf\_type & \_\_sb, char\_type \_\_delim)
- \_\_istream\_type & [get](#) (\_\_streambuf\_type & \_\_sb)
- \_\_istream\_type & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- \_\_istream\_type & [getline](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
- \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n)
- \_\_istream\_type & [ignore](#) ()
- \_\_istream\_type & [ignore](#) ([streamsize](#) \_\_n, char \_\_delim)
- int\_type [peek](#) ()
- \_\_istream\_type & [read](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- \_\_istream\_type & [putback](#) (char\_type \_\_c)
- \_\_istream\_type & [unget](#) ()
- int [sync](#) ()
- pos\_type [tellg](#) ()

- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator bool () const`
- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Static Public Attributes

- static const [openmode](#) `__noreplace`
- static const [fmtflags](#) `adjustfield`
- static const [openmode](#) `app`
- static const [openmode](#) `ate`
- static const [iostate](#) `badbit`
- static const [fmtflags](#) `basefield`
- static const [seekdir](#) `beg`
- static const [openmode](#) `binary`
- static const [fmtflags](#) `boolalpha`
- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_cache\\_locale](#) (const [locale](#) &\_\_loc)
- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- template<typename \_ValueT>  
    \_\_istream\_type & [\\_M\\_extract](#) (\_ValueT &\_\_v)
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- template<typename \_ValueT>  
    \_\_ostream\_type & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept

- void **init** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_iostream](#) &\_\_rhs)
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)

### Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

### 6.254.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringstream< _CharT, _Traits, _Alloc >
```

Controlling input and output for `std::string`.

### Template Parameters

|                         |                                                                                 |
|-------------------------|---------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                       |
| <a href="#">_Traits</a> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

### 6.254.2 Member Typedef Documentation

#### event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                                                                                                    |                                                        |
|----------------------------------------------------------------------------------------------------|--------------------------------------------------------|
|  <code>__e</code> | One of the members of the event enum.                  |
|  <code>__b</code> | Reference to the ios_base object.                      |
|  <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

#### fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`

- baselfield
- floatfield

### **iostate**

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

### **openmode**

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

### **seekdir**

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

## **6.254.3 Member Enumeration Documentation**

### **event**

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.



#### 6.254.4 Constructor & Destructor Documentation

##### basic\_stringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using the mode `in|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

##### basic\_stringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
 ios_base::openmode __m) [inline], [explicit]
```

Starts with an empty string buffer.

##### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__m</code> | Whether the buffer can read, or write, or both. |
|------------------|-------------------------------------------------|

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

##### basic\_stringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
 const __string_type & __str,
 ios_base::openmode __m = ios_base::out | ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

##### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__str</code> | A string to copy as a starting buffer.          |
| <code>__m</code>   | Whether the buffer can read, or write, or both. |

Initializes `sb` using `__str` and `__m`, and passes `&sb` to the base class initializer. That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

##### ~basic\_stringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::~~basic_stringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

#### 6.254.5 Member Function Documentation

##### \_M\_getloc()

```
const locale & std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

**bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other `iostate` flags may also be set.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**clear()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

References `std::ios_base::badbit`, `exceptions()`, `rdbuf()`, and `rdstate()`.

Referenced by `std::basic_ios<char_type, traits_type>::exceptions()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::__detail::operator>>()`, `std::basic_istream<_CharT, _Traits>::putback()`, `rdbuf()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ios<char_type, traits_type>::seekg()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

**copyfmt()**

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

References [basic\\_ios\(\)](#), [std::\\_\\_addressof\(\)](#), [exceptions\(\)](#), [fill\(\)](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::getloc\(\)](#), [std::ios\\_base::precision\(\)](#), [tie\(\)](#), [std::tie\(\)](#), and [std::ios\\_base::width\(\)](#).

**eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof() const [inline], [nodiscard], [inherited]
Fast error checking.
```

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

**exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline], [nodiscard], [inherited]
Throwing exceptions on errors.
```

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Referenced by [clear\(\)](#), and [copyfmt\(\)](#).

**exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::exceptions (
 iostate __except) [inline], [inherited]
Throwing exceptions on errors.
```

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

 std::ifstream f ("/etc/motd");

 std::cerr << "Setting badbit\n";
```

```
f.setstate (std::ios_base::badbit);

std::cerr « "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}
```

### fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [nodiscard], [inherited]
Fast error checking.
```

#### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Referenced by [std::basic\\_ios< char\\_type, traits\\_type >::operator bool\(\)](#), [std::basic\\_ios< char\\_type, traits\\_type >::operator!\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_ostream< \\_CharT, \\_Traits >::seekp\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

### fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [nodiscard], [inherited]
Retrieves the empty character.
```

#### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Referenced by [copyfmt\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator bool\(\)](#).

### fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

#### Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

### flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard], [inherited]
Access to format flags.
```

**Returns**

The format control flags for both input and output.

Referenced by [std::basic\\_ios<\\_CharT, \\_Traits>::copyfmt\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_get<\\_CharT, \\_InIter>::do\\_get\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::num\\_put<\\_CharT, \\_OutIter>::do\\_put\(\)](#), [std::chrono::operator<<\(\)](#), [std::philoX\\_engine<uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<<\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::philoX\\_engine<uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>\(\)](#).

**flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

**flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::flush\(\)](#).

Referenced by [std::flush\(\)](#).

**gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

**get()** [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#), and [std::basic\\_istream<char\\_type, traits\\_type>::get\(\)](#).

**get()** [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb, widen("\n"))`.

**get()** [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::](#)

**get()** [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                          |                                       |
|--------------------------|---------------------------------------|
| $\leftrightarrow$<br>__c | The character in which to store data. |
|--------------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**get()** [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                          |                                             |
|--------------------------|---------------------------------------------|
| $\leftrightarrow$<br>__s | Pointer to an array.                        |
| $\leftrightarrow$<br>__n | Maximum number of characters to store in s. |

**Returns**

\*this

Returns [get\(\\_\\_s, \\_\\_n, widen\("\n"\)\)](#).

**get()** [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

References [\\_M\\_gcount](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::sgetc\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::snextc\(\)](#).

**getline() [1/3]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                        |                                               |
|------------------------|-----------------------------------------------|
| <code>↵<br/>__s</code> | A character array in which to store the data. |
| <code>↵<br/>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

**getline() [2/3]**

```
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.



**getline()** [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

References `_M_gcount`, and `std::ios_base::goodbit`.

Referenced by `std::basic_istream< char_type, traits_type >::getline()`.

**getloc()**

```
locale std::ios_base::getloc () const [inline], [nodiscard], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::operator<<()`, `std::operator<>()`, `std::operator>>()`, and `std::ws()`.

**good()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [nodiscard], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [std::\\_\\_detail::operator>>\(\)](#).

**ignore() [1/4]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

Referenced by [std::basic\\_istream<char\\_type, traits\\_type>::ignore\(\)](#).

**ignore() [2/4]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

References [\\_M\\_gcount](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sbumpc\(\)](#).

**ignore() [3/4]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 char __delim) [inline], [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

**ignore() [4/4]**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.



```
std::basic_istream< char >::basic_istream(), std::basic_istream< char >::basic_istream(),
std::basic_istream< char >::basic_istream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostream< char >::basic_ostream(), std::basic_ostream< char >::basic_ostream(),
std::basic_ostream< char >::basic_ostream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().
```

## iword()

```
long & std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [iword\(\)](#).

Referenced by [iword\(\)](#).

## narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Squeezes characters.

### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

**operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator"!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

**operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 __ios_type &(* __pf) (__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [2/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 __ostream_type &(* __pf) (__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

**operator<<() [3/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

**operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|       |                             |
|-------|-----------------------------|
| $\_p$ | A variable of pointer type. |
|-------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $\_f$ | A variable of builtin floating point type. |
|-------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<()** [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|              |                                            |
|--------------|--------------------------------------------|
| $\leftarrow$ | A variable of builtin floating point type. |
| $\leftarrow$ |                                            |
| $\leftarrow$ |                                            |
| $\leftarrow$ |                                            |
| $f$          |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| $\leftarrow$ | A variable of builtin integral type. |
| $n$          |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to perform numeric formatting.

**operator<<()** [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(* __pf)(ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::endl and std::hex use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

**operator<<()** [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                               |                                      |
|-------------------------------|--------------------------------------|
| $\_ \leftrightarrow$<br>$\_n$ | A variable of builtin integral type. |
|-------------------------------|--------------------------------------|

**Returns**

$\ast this$  if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                                                                  |                                            |
|--------------------------------------------------------------------------------------------------|--------------------------------------------|
| $\leftrightarrow$<br>$\_ \leftrightarrow$<br>$\_ \leftrightarrow$<br>$\_ \leftrightarrow$<br>$f$ | A variable of builtin floating point type. |
|--------------------------------------------------------------------------------------------------|--------------------------------------------|

**Returns**

$\ast this$  if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                               |                                      |
|-------------------------------|--------------------------------------|
| $\_ \leftrightarrow$<br>$\_n$ | A variable of builtin integral type. |
|-------------------------------|--------------------------------------|

**Returns**

$\ast this$  if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

**operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```



Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

References [basic\\_ostream\(\)](#), [std::ios\\_base::badbit](#), [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>::failed\(\)](#), [std::ios\\_base::goodbit](#), [std::num\\_put<\\_CharT, \\_Outiter>::put\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::use\\_facet\(\)](#).

#### `operator<<()` [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

#### Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### `operator<<()` [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_x$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator<<()** [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

#### Parameters

|                |                                      |
|----------------|--------------------------------------|
| $\_x$<br>$\_n$ | A variable of builtin integral type. |
|----------------|--------------------------------------|

#### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

#### **operator>>()** [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(* __pf)(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

#### **operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(* __pf)(__istream_type &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

#### **operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

### Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

References [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

### `operator>>()` [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

### Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

### `operator>>()` [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

### Parameters

|                |                                            |
|----------------|--------------------------------------------|
| <code>f</code> | A variable of builtin floating point type. |
|----------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|          |                                      |
|----------|--------------------------------------|
| ↵        | A variable of builtin integral type. |
| <i>n</i> |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>()** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base & (* __pf) (ios_base &)) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

**operator>>()** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|              |                                            |
|--------------|--------------------------------------------|
| $\leftarrow$ | A variable of builtin floating point type. |
| $\leftarrow$ |                                            |
| $\leftarrow$ |                                            |
| $\leftarrow$ |                                            |
| $f$          |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>()** [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                |                                      |
|----------------|--------------------------------------|
| $\leftarrow$   | A variable of builtin integral type. |
| $\leftarrow n$ |                                      |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                    |                                      |
|----------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|----------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

References [std::ios\\_base::badbit](#), [std::ios\\_base::failbit](#), [std::num\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::use\\_facet\(\)](#).

**operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|------------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**operator>>() [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [15/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [16/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| $\leftrightarrow$<br>__n | A variable of builtin integral type. |
|--------------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

**operator>>() [17/17]**

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream<_CharT, _Traits>::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↔<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

**peek()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

**precision() [1/2]**

```
streamsize std::ios_base::precision () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic\\_ios< \\_CharT, \\_Traits >::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

**precision() [2/2]**

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

**put()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```



Simple insertion.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

#### Returns

`*this`

Tries to insert `__c`.

#### Note

This function is not overloaded on signed char and unsigned char.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::putback\(\)](#).  
Referenced by [std::endl\(\)](#), and [std::ends\(\)](#).

#### putback()

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

#### Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

#### Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sputbackc\(\)](#).  
Referenced by [std::operator<<\(\)](#).

#### pword()

```
void *& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

### Parameters

|                  |                       |
|------------------|-----------------------|
| <code>_ix</code> | Index into the array. |
|------------------|-----------------------|

### Returns

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

### `rdbuf()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

### Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

### Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;

foo.ios::rdbuf(p); // ios == basic_ios<char>
```

References [clear\(\)](#).

### `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf () const [inline],
[nodiscard]
```

Accessing the underlying buffer.

### Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

### `rdstate()`

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Referenced by `std::basic_ios<char_type, traits_type>::bad()`, `clear()`, `std::basic_ios<char_type, traits_type>::eof()`, `std::basic_ios<char_type, traits_type>::fail()`, `std::basic_ios<char_type, traits_type>::good()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::unget()`.

**read()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

References `_M_gcount`.

**readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::min\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**seekg() [1/2]**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets `failbit`.

**Note**

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::ios\\_base::in](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>](#) and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

References [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**seekp()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

References [std::basic\\_ios< \\_CharT, \\_Traits >::fail\(\)](#), [std::ios\\_base::failbit](#), [std::ios\\_base::out](#), [std::basic\\_ios< \\_CharT, \\_Traits >::rdbuf\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#).

**setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

**setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, `std::basic_fstream<char>::close()`, `std::basic_ifstream<char>::close()`, `std::basic_ofstream<char>::close()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::getline()`, `std::getline()`, `std::basic_fstream<char>::open()`, `std::basic_fstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ifstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ofstream<char>::open()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, `std::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::tr2::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_ostream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

**str() [1/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringstream<_CharT, _Traits, _Alloc>::str () const & [inline], [nodiscard]
```

Copying out the string buffer.

**Returns**

```
rdbuf()->str()
```

**str() [2/2]**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringstream<_CharT, _Traits, _Alloc>::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf()->str(s)`.

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References [std::ios\\_base::badbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_streambuf<\\_CharT, \\_Traits>::pubsync\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#).

**sync\_with\_stdio()**

```
bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() ->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

References [std::ios\\_base::badbit](#), [std::ios\\_base::cur](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::in](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp () [inherited]
```

Getting the current write position.



**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

References [std::ios\\_base::cur](#), [std::basic\\_ios<\\_CharT, \\_Traits>::fail\(\)](#), [std::ios\\_base::out](#), and [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#).

**tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie () const [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry::sentry\(\)](#), and [copyfmt\(\)](#).

**tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

**unget()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References [\\_M\\_gcount](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::clear\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdbuf\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::rdstate\(\)](#), [std::basic\\_ios<\\_CharT, \\_Traits>::setstate\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits>::sungetc\(\)](#).

Referenced by [std::\\_\\_detail::operator>>\(\)](#).

**unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

**widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Referenced by `std::endl()`, `std::basic_ios< char_type, traits_type >::fill()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< char_type, traits_type >::getline()`, `std::getline()`, `std::getline()`, `std::operator<<()`, `std::operator<<()`, `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F` and `std::tr2::operator>>()`.

**width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::operator>>()`, and `std::operator>>()`.

**width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

### Returns

The previous value of `width()`.

### write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

### Returns

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

### Note

This function is not overloaded on signed char and unsigned char.

### xalloc()

```
int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 6.254.6 Member Data Documentation

### **`_M_gcount`**

```
template<typename _CharT, typename _Traits>
```

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Referenced by `get()`, `get()`, `get()`, `get()`, `getline()`, `ignore()`, `ignore()`, `ignore()`, `putback()`, `read()`, `readsome()`, and `unget()`.

### **`adjustfield`**

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### **`app`**

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsputn()`.

### **`ate`**

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

### **`badbit`**

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_ostream< _CharT, _Traits >::sentry::~~sentry()`, `std::basic_ios< _CharT, _Traits >::clear()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_iostream< char_type, traits_type >::operator<<()`, `std::basic_iostream< char_type, traits_type >::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

### **`basefield`**

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::oct()`.

### **`beg`**

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

**binary**

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↔filestreams.binary>.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::showmanyc\(\)](#).

**boolalpha**

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract bool in alphabetic rather than numeric format.

Referenced by [std::boolalpha\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_put< \\_CharT, \\_OutIter >::do\\_put\(\)](#), and [std::noboolalpha\(\)](#).

**cur**

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::imbue\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::tellg\(\)](#), and [std::basic\\_ostream< \\_CharT, \\_Traits >::tellp\(\)](#).

**dec**

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by [std::dec\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>\(\)](#), and [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<\(\)](#).

**end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::open\(\)](#), and [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#).

**eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::num\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_monthname\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_time\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::basic\\_iostream< char\\_type, traits\\_type >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::get\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::ignore\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::putback\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::readsome\(\)](#), [std::basic\\_iostream< char\\_type, traits\\_type >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::seekg\(\)](#), [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#), and [std::ws\(\)](#).

**failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.



**internal**

Referenced by [std::internal\(\)](#).

Referenced by `std::num_put<_CharT, _Outiter>::do_put()`, `std::left()`, and `std::phlox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0`.

Referenced by [std::oct\(\)](#).

Referenced by std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_fstream(), std::basic\_fstream< char >::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ofstream< char >::basic\_ofstream(), std::basic\_ostringstream< char >::basic\_ostringstream(), std::basic\_stringbuf< char >::basic\_stringbuf(), std::basic\_stringstream< char >::basic\_stringstream(), std::basic\_filebuf< char\_type, traits\_type >::M\_set\_buffer(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_fstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_ofstream< char >::open(), std::basic\_filebuf< \_CharT, Traits >::overflow(), std::basic\_stringbuf< \_CharT, Traits, Alloc >::overflow(), std::basic\_stringbuf< \_CharT, Traits, Alloc >::ebackfail(), std::basic\_streambuf< char\_type, traits\_type >::pubseekoff(), std::basic\_streambuf< char\_type, traits\_type >::pubseekpos(), std::basic\_streambuf< char\_type, traits\_type >::seekoff(), std::basic\_stringbuf< \_CharT, Traits, Alloc >::seekoff(), std::basic\_stringbuf< char >::seekoff(), std::basic\_ostream< \_CharT, Traits >::seekp(), std::basic\_ostream< \_CharT, Traits >::seekp(), std::basic\_streambuf< char\_type, traits\_type >::seekpos(), std::basic\_stringbuf< \_CharT, Traits, Alloc >::seekpos(), std::basic\_stringbuf< char >::seekpos(), std::basic\_ostream< \_CharT, Traits >::tellp(), and std::basic\_filebuf< \_CharT, Traits >::xsputn.

Referenced by [std::right\(\)](#).

Referenced by `std::hexfloat()`, and `std::scientific()`.

**showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

**showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

**showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

**skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::\\_\\_detail::operator>>\(\)](#), [std::philox\\_engine< uint\\_fast32\\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x1E3779B9, 0x1E3779B9 >::operator>>\(\)](#), and [std::skipws\(\)](#).

**trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

**unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::sentry::~sentry\(\)](#), [std::nunitbuf\(\)](#), and [std::unitbuf\(\)](#).

**uppercase**

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [sstream](#)

**6.255 std::bernoulli\_distribution Class Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef bool [result\\_type](#)



**Public Member Functions**

- [bernoulli\\_distribution](#) ()
- [bernoulli\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [bernoulli\\_distribution](#) (double \_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

**Friends**

- bool [operator==](#) (const [bernoulli\\_distribution](#) &\_\_d1, const [bernoulli\\_distribution](#) &\_\_d2)

**6.255.1 Detailed Description**

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Since

C++11

**6.255.2 Member Typedef Documentation****result\_type**

```
typedef bool std::bernoulli_distribution::result_type
```

The type of the range of the distribution.

**6.255.3 Constructor & Destructor Documentation****bernoulli\_distribution()** [1/2]

```
std::bernoulli_distribution::bernoulli_distribution () [inline]
```

Constructs a Bernoulli distribution with likelihood 0.5.

References [bernoulli\\_distribution\(\)](#).

Referenced by [bernoulli\\_distribution\(\)](#), and [operator==](#).

**bernoulli\_distribution()** [2/2]

```
std::bernoulli_distribution::bernoulli_distribution (
 double __p) [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood *p*.

**Parameters**

|                  |                                                                                      |
|------------------|--------------------------------------------------------------------------------------|
| <code>__p</code> | [IN] The likelihood of a true result being returned. Must be in the interval [0, 1]. |
|------------------|--------------------------------------------------------------------------------------|

**6.255.4 Member Function Documentation****max()**

```
result_type std::bernoulli_distribution::max () const [inline]
```

Returns the least upper bound value of the distribution.

References [std::numeric\\_limits<\\_Tp>::max\(\)](#).

**min()**

```
result_type std::bernoulli_distribution::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

References [std::numeric\\_limits<\\_Tp>::min\(\)](#).

**operator()()**

```
template<typename _UniformRandomNumberGenerator>
result_type std::bernoulli_distribution::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

References [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

**p()**

```
double std::bernoulli_distribution::p () const [inline]
```

Returns the *p* parameter of the distribution.

**param()** [1/2]

```
param_type std::bernoulli_distribution::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param()** [2/2]

```
void std::bernoulli_distribution::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
void std::bernoulli_distribution::reset () [inline]
```

Resets the distribution state.

Does nothing for a Bernoulli distribution.

**6.255.5 Friends And Related Symbol Documentation****operator==**

```
bool operator== (
 const bernoulli_distribution & __d1,
 const bernoulli_distribution & __d2) [friend]
```

Return true if two Bernoulli distributions have the same parameters.

References [bernoulli\\_distribution\(\)](#).

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.256 std::bidirectional\_iterator\_tag Struct Reference**

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::bidirectional\_iterator\_tag:



### 6.256.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.  
The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 6.257 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

```
#include <point_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



### Public Types

- typedef `Const_Pointer` **const\_pointer**
- typedef `Const_Reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**

- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

### Public Member Functions

- **bin\_search\_tree\_const\_it** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- **bin\_search\_tree\_const\_it** (const Node\_Pointer p\_nd=0)
- bool **operator!=** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator!=** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- const\_reference **operator\*** () const
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator++** ()
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator++** (int)
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator--** ()
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator--** (int)
- const\_pointer **operator->** () const
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator=** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator=** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- bool **operator==** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator==** (const bin\_search\_tree\_const\_it < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const

### Public Attributes

- Node\_Pointer **m\_p\_nd**

### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 6.257.1 Detailed Description

**template**<typename Node\_Pointer, typename Value\_Type, typename Pointer, typename Const\_Pointer, typename Reference, typename Const\_Reference, bool Is\_Forward\_Iterator, typename \_Alloc>  
**class** `__gnu_pbds::detail::bin_search_tree_const_it` < Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >

Const iterator.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

## 6.258 `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



### Public Types

- typedef `Const_Iterator` [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) `difference_type`
- typedef [trivial\\_iterator\\_tag](#) `iterator_category`
- typedef [rebind\\_traits](#)< `_Alloc`, [metadata\\_type](#) >::`const_reference` `metadata_const_reference`
- typedef `Node::metadata_type` [metadata\\_type](#)
- typedef `Const_Iterator` [reference](#)
- typedef `Const_Iterator` [value\\_type](#)

### Public Member Functions

- `bin_search_tree_const_node_it_` (`const node_pointer p_nd=0`)
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` [get\\_r\\_child](#) () const
- bool [operator!=](#) (`const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other`) const
- [const\\_reference](#) [operator\\*](#) () const
- bool [operator==](#) (`const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other`) const

### Public Attributes

- node\_pointer `m_p_nd`

#### 6.258.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

## 6.258.2 Member Typedef Documentation

### `const_reference`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::const_reference
```

Iterator's `__const` reference type.

### `difference_type`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type
```

Difference type.

### `iterator_category`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↵
Iterator, Iterator, _Alloc >::iterator_category
```

Category.

### `metadata_const_reference`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

### `metadata_type`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↵
Iterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

### `reference`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::reference
```

Iterator's reference type.

### `value_type`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
```

Iterator's value type.

## 6.258.3 Member Function Documentation

### `get_l_child()`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
```

```
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
```

Returns the `__const` node iterator associated with the left node.

### **get\_metadata()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline]
```

Metadata access.

### **get\_r\_child()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
```

Returns the `__const` node iterator associated with the right node.

### **operator!=()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline]
```

Compares (negatively) to a different iterator object.

### **operator\*()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* () const [inline]
```

Access.

### **operator==()**

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## **6.259 \_\_gnu\_pbds::detail::bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > Class Template Reference**

```
#include <point_iterators.hpp>
```



Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



## Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- **bin\_search\_tree\_it\_** (const bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- **bin\_search\_tree\_it\_** (const Node\_Pointer p\_nd=0)
- bool **operator!=** (const bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator!=** (const bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::reference **operator\*** () const
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator++** ()
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator++** (int)
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator--** ()
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **operator--** (int)
- bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >::pointer **operator->** () const
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator=** (const bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > & **operator=** (const bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)

- bool **operator==** (const bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator==** (const bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const

#### Public Attributes

- Node\_Pointer **m\_p\_nd**

#### Protected Types

- typedef bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > **base\_it\_type**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 6.259.1 Detailed Description

template<typename Node\_Pointer, typename Value\_Type, typename Pointer, typename Const\_Pointer, typename Reference, typename Const\_Reference, bool Is\_Forward\_Iterator, typename \_Alloc>  
class \_\_gnu\_pbds::detail::bin\_search\_tree\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >

Iterator.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 6.260 \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc > Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >:



## Public Types

- typedef Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits](#)< [\\_Alloc](#), [metadata\\_type](#) >::[const\\_reference](#) [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value\\_type](#)

## Public Member Functions

- [bin\\_search\\_tree\\_node\\_it\\_](#) (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const
- Iterator [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const

## Public Attributes

- node\_pointer [m\\_p\\_nd](#)

### 6.260.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Node iterator.

### 6.260.2 Member Typedef Documentation

#### [const\\_reference](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↵
_Alloc >::const_reference
Iterator's __const reference type.
```

#### [difference\\_type](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type [inherited]
Difference type.
```

#### [iterator\\_category](#)

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↵
Iterator, Iterator, _Alloc >::iterator_category [inherited]
Category.
```

### metadata\_const\_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
Const metadata reference type.
```

### metadata\_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator,
Iterator, Iterator, _Alloc >::metadata_type [inherited]
Metadata type.
```

### reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >::reference
Iterator's reference type.
```

### value\_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >::value_type
Iterator's value type.
```

## 6.260.3 Member Function Documentation

### get\_l\_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
Returns the node iterator associated with the left node.
```

### get\_metadata()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline], [inherited]
Metadata access.
```

### get\_r\_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
Returns the node iterator associated with the right node.
```

### operator"!="()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
```

```
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline], [inherited]
```

Compares (negatively) to a different iterator object.

### `operator*()`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator* () const [inline]
```

Access.

### `operator==( )`

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator==(
```

```
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline], [inherited]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

## 6.261 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node` **update\_pointer**
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 6.261.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr,
class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

## 6.261.2 Member Typedef Documentation

### node\_const\_iterator

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 6.262 [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#)< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc > Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef [bin\\_search\\_tree\\_const\\_it\\_](#)< typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef Node **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it\\_](#)< Node, point\_const\_iterator, point\_iterator, \_Alloc > [node\\_const\\_iterator](#)
- typedef [node\\_const\\_iterator](#) **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), node\_iterator, Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#)< [node\\_const\\_iterator](#), node\_iterator, Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it\\_](#)< typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef point\_const\_iterator **point\_iterator**
- typedef const\_reverse\_iterator **reverse\_iterator**

### 6.262.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
```

Specialization.

## 6.262.2 Member Typedef Documentation

### node\_const\_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 6.263 `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >` Class Template Reference

```
#include <functional>
```

Inheritance diagram for `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`:



### Public Types

- typedef `_Operation2::argument_type` [argument\\_type](#)
- typedef `_Operation1::result_type` [result\\_type](#)

### Public Member Functions

- **binary\_compose** (const `_Operation1` &`_x`, const `_Operation2` &`_y`, const `_Operation3` &`_z`)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &`_x`) const

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**
- `_Operation3` **\_M\_fn3**

#### 6.263.1 Detailed Description

```
template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
```

An [SGI extension](#) .

#### 6.263.2 Member Typedef Documentation

##### `argument_type`

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1::result_type >::argument_type [inherited]
```

`argument_type` is the type of the argument

**result\_type**

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1::↵
result_type >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [ext/functional](#)

**6.264 std::binary\_function< \_Arg1, \_Arg2, \_Result > Struct Template Reference**

```
#include <stl_function.h>
```



Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### 6.264.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >
```

Helper for defining adaptable binary function objects.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17.

### 6.264.2 Member Typedef Documentation

#### first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type
first_argument_type is the type of the first argument
```

#### result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
result_type is the return type
```

#### second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.265 \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <binary_heap.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond\_dealtor\_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `rebind_traits< _Alloc, entry >::allocator_type` **entry\_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry\_cmp**
- typedef `rebind_traits< _Alloc, entry >::pointer` **entry\_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binary\_heap** (const `binary_heap` &)
- **binary\_heap** (const `cmp_fn` &)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- void **erase\_at** (`entry_pointer`, `size_type`, `false_type`)
- void **erase\_at** (`entry_pointer`, `size_type`, `true_type`)
- template<typename `Pred`>  
  `size_type` **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (`size_type`) const
- void **join** (`binary_heap` &)
- `size_type` **max\_size** () const
- void **modify** (`point_iterator`, `const_reference`)
- void **notify\_arbitrary** (`size_type`)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- void **pop** ()
- `point_iterator` **push** (`const_reference`)
- bool **resize\_needed\_for\_grow** (`size_type`) const

- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, binary\_heap &)
- void **swap** (binary\_heap &)
- void **swap** (resize\_policy< \_Alloc::size\_type > &)
- const\_reference **top** () const

### Static Public Attributes

- static const \_Alloc::size\_type **min\_size**

### Protected Member Functions

- template<typename It>  
void **copy\_from\_range** (It, It)

#### 6.265.1 Detailed Description

template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>  
class **\_\_gnu\_pbds::detail::binary\_heap**< Value\_Type, Cmp\_Fn, \_Alloc >

Binary heaps composed of resize and compare policies.  
Based on CLRS.

The documentation for this class was generated from the following file:

- [binary\\_heap.hpp](#)

#### 6.266 **\_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc > Class Template Reference**

#include <const\_iterator.hpp>

Inheritance diagram for **\_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >**:



### Public Types

- typedef [base\\_type::const\\_pointer](#) **const\_pointer**
- typedef [base\\_type::const\\_reference](#) **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**

- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

### Public Member Functions

- `binary_heap_const_iterator_()`
- `binary_heap_const_iterator_(const binary_heap_const_iterator_ &other)`
- `binary_heap_const_iterator_(entry_pointer p_e)`
- `bool operator!= (const binary_heap_const_iterator_ &other) const`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `binary_heap_const_iterator_ & operator++ ()`
- `binary_heap_const_iterator_ operator++ (int)`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_const_iterator_ &other) const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

### Public Attributes

- `entry_pointer m_p_e`

#### 6.266.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

#### 6.266.2 Member Typedef Documentation

##### `const_pointer`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_pointer
Iterator's const pointer type.
```

##### `const_reference`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_reference
Iterator's const reference type.
```

##### `difference_type`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type
Difference type.
```

### iterator\_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
Category.
```

### pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::pointer
Iterator's pointer type.
```

### reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::reference
Iterator's reference type.
```

### value\_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
Iterator's value type.
```

## 6.266.3 Constructor & Destructor Documentation

### binary\_heap\_const\_iterator\_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↵
heap_const_iterator_ () [inline]
Default constructor.
```

### binary\_heap\_const\_iterator\_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↵
heap_const_iterator_ (
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
Copy constructor.
```

## 6.266.4 Member Function Documentation

### operator"!="() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↵
operator!= (
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
Compares content (negatively) to a different iterator object.
```

### `operator!=()` [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator!= (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

### `operator*()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline], [inherited]
```

Access.

### `operator->()`

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

### `operator==()` [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↵
operator== (
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

### `operator==()` [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator== (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/const\\_iterator.hpp](#)

## 6.267 `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc`

&gt;:



## Public Types

- typedef `rebind_traits< _Alloc, value_type >::const_pointer` `const_pointer`
- typedef `rebind_traits< _Alloc, value_type >::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `rebind_traits< _Alloc, value_type >::pointer` `pointer`
- typedef `rebind_traits< _Alloc, value_type >::reference` `reference`
- typedef `Value_Type` `value_type`

## Public Member Functions

- `binary_heap_point_const_iterator_()`
- `binary_heap_point_const_iterator_(const binary_heap_point_const_iterator_ &other)`
- `binary_heap_point_const_iterator_(entry_pointer p_e)`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

## Public Attributes

- entry\_pointer `m_p_e`

## Protected Types

- typedef `rebind_traits< _Alloc, Entry >::pointer` `entry_pointer`



### 6.267.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

### 6.267.2 Member Typedef Documentation

#### **const\_pointer**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc, value_type>::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_pointer
Iterator's const pointer type.
```

#### **const\_reference**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc, value_type>::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_reference
Iterator's const reference type.
```

#### **difference\_type**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::difference_type
Difference type.
```

#### **iterator\_category**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
Category.
```

#### **pointer**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc, value_type>::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::pointer
Iterator's pointer type.
```

#### **reference**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc, value_type>::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference
Iterator's reference type.
```

#### **value\_type**

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
Iterator's value type.
```

### 6.267.3 Constructor & Destructor Documentation

#### binary\_heap\_point\_const\_iterator\_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↵
binary_heap_point_const_iterator_ () [inline]
```

Default constructor.

#### binary\_heap\_point\_const\_iterator\_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↵
binary_heap_point_const_iterator_ (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```

Copy constructor.

### 6.267.4 Member Function Documentation

#### operator"!="()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator!= (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

#### operator\*()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline]
```

Access.

#### operator->()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> () const [inline]
```

Access.

#### operator==(())

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator==(
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_point\\_const\\_iterator.hpp](#)

**6.268 \_\_gnu\_pbds::binary\_heap\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::binary\_heap\_tag:

**6.268.1 Detailed Description**

Binary-heap (array-based).

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

**6.269 std::binary\_negate<\_Predicate> Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::binary_negate<_Predicate>`:

**Public Types**

- `typedef _Predicate::first_argument_type` [first\\_argument\\_type](#)

- typedef bool [result\\_type](#)
- typedef [\\_Predicate::second\\_argument\\_type](#) [second\\_argument\\_type](#)

### Public Member Functions

- constexpr **binary\_negate** (const [\\_Predicate](#) &\_\_x)
- constexpr bool **operator()** (const typename [\\_Predicate::first\\_argument\\_type](#) &\_\_x, const typename [\\_Predicate::second\\_argument\\_type](#) &\_\_y) const

### Protected Attributes

- [\\_Predicate\\_M\\_pred](#)

## 6.269.1 Detailed Description

```
template<typename _Predicate>
class std::binary_negate< _Predicate >
```

One of the [negation functors](#).

## 6.269.2 Member Typedef Documentation

### [first\\_argument\\_type](#)

```
typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

### [result\\_type](#)

```
typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]
result_type is the return type
```

### [second\\_argument\\_type](#)

```
typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 6.270 [std::binder1st<\\_Operation>](#) Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for std::binder1st< \_Operation >:



### Public Types

- typedef `_Operation::second_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

### Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

### Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

#### 6.270.1 Detailed Description

```
template<typename _Operation>
class std::binder1st< _Operation >
```

One of the [binder functors](#).

#### 6.270.2 Member Typedef Documentation

##### `argument_type`

```
typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type, ↔
_Operation::result_type >::argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef _Operation::result_type std::unary_function< _Operation::second_argument_type, _Operation::↵
result_type >::result_type [inherited]
```

result\_type is the return type

The documentation for this class was generated from the following file:

- [backward/binders.h](#)

**6.271 std::binder2nd< \_Operation > Class Template Reference**

```
#include <binders.h>
```

Inheritance diagram for std::binder2nd< \_Operation >:

**Public Types**

- typedef `_Operation::first_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

**Public Member Functions**

- **binder2nd** (const `_Operation` &\_\_x, const typename `_Operation::second_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::first_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::first_argument_type` &\_\_x) const

**Protected Attributes**

- `_Operation` **op**
- `_Operation::second_argument_type` **value**

**6.271.1 Detailed Description**

```
template<typename _Operation>
class std::binder2nd< _Operation >
```

One of the [binder functors](#).

### 6.271.2 Member Typedef Documentation

#### argument\_type

typedef `_Operation::first_argument_type` `std::unary_function<_Operation::first_argument_type, _Operation::result_type>::argument_type` [inherited]  
`argument_type` is the type of the argument

#### result\_type

typedef `_Operation::result_type` `std::unary_function<_Operation::first_argument_type, _Operation::result_type>::result_type` [inherited]  
`result_type` is the return type

The documentation for this class was generated from the following file:

- [backward/binders.h](#)

## 6.272 `std::binomial_distribution<_IntType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- **binomial\_distribution** (`_IntType` \_\_t, double \_\_p=0.5)
- **binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- `_IntType` **t** () const

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::binomial_distribution< _IntType1 > &__x)`
- `bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::binomial_distribution< _IntType1 > &__x)`

### 6.272.1 Detailed Description

```
template<typename _IntType = int>
class std::binomial_distribution< _IntType >
```

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Since

C++11

### 6.272.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 6.272.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator()() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::binomial_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by `operator()()`.



**operator()()** [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_FUNCS` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

**p()**

```
template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p () const [inline]
```

Returns the distribution `p` parameter.

**param()** [1/2]

```
template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param()** [2/2]

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

**t()**

```
template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t () const [inline]
```

Returns the distribution `t` parameter.

**6.272.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::binomial_distribution< _IntType1 > & __x) [friend]
```

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>binomial_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### `operator==`

```
template<typename _IntType = int>
bool operator== (
 const binomial_distribution< _IntType > & __d1,
 const binomial_distribution< _IntType > & __d2) [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

#### `operator>>`

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::binomial_distribution< _IntType1 > & __x) [friend]
```

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

#### Parameters

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                     |
| <code>__x</code>  | A <code>binomial_distribution</code> random number generator engine. |

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.273 `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <binomial_heap.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `base_type::allocator_type` **allocator\_type**
- typedef `base_type::cmp_fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binomial\_heap** (const `binomial_heap` &)
- **binomial\_heap** (const `Cmp_Fn` &)
- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point\_iterator)
- template<typename `Pred`>  
size\_type **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- void **join** (`binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename `Pred`>  
void **split** (`Pred`, `binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `_Alloc::size_type`, `_Alloc` > &)
- const\_reference **top** () const

### Protected Types

- typedef base\_type::node **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 6.273.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binomial heap.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_.hpp](#)

#### 6.274 \_\_gnu\_pbds::detail::binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <binomial_heap_base_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- void **join** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()

- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef base\_type::node **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef base\_type::node\_const\_pointer **node\_const\_pointer**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- **binomial\_heap\_base** (const binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- **binomial\_heap\_base** (const Cmp\_Fn &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 6.274.1 Detailed Description

**template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>**  
**class** **\_\_gnu\_pbds::detail::binomial\_heap\_base**< Value\_Type, Cmp\_Fn, \_Alloc >

Base class for binomial heap.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_base\\_.hpp](#)

## 6.275 \_\_gnu\_pbds::binomial\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::binomial\_heap\_tag:



### 6.275.1 Detailed Description

Binomial-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.276 \_\_gnu\_cxx::bitmap\_allocator<\_Tp> Class Template Reference

```
#include <bitmap_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::bitmap\_allocator<\_Tp>:



## Public Types

- typedef free\_list::\_\_mutex\_type **\_\_mutex\_type**
- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **bitmap\_allocator** (const bitmap\_allocator &) noexcept
- template<typename \_Tp1>  
**bitmap\_allocator** (const bitmap\_allocator< \_Tp1 > &) noexcept
- pointer **\_M\_allocate\_single\_object** ()
- void **\_M\_deallocate\_single\_object** (pointer \_\_p) throw ()
- const\_pointer **address** (const\_reference \_\_r) const noexcept
- pointer **address** (reference \_\_r) const noexcept
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, typename bitmap\_allocator< void >::const\_pointer)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) throw ()
- template<typename \_Up>  
void **destroy** (\_Up \* \_\_p)
- size\_type **max\_size** () const noexcept

### 6.276.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::bitmap_allocator< _Tp >
```

Bitmap Allocator, primary template.

### 6.276.2 Member Function Documentation

#### **\_M\_allocate\_single\_object()**

```
template<typename _Tp>
pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object () [inline]
Allocates memory for a single object of size sizeof(_Tp).
```

## Exceptions

|                        |                                |
|------------------------|--------------------------------|
| <i>std::bad_alloc.</i> | If memory cannot be allocated. |
|------------------------|--------------------------------|

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.



**\_M\_deallocate\_single\_object()**

```
template<typename _Tp>
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
 pointer __p) throw () [inline]
```

Deallocates memory that belongs to a single object of size sizeof(\_Tp).

Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

**6.277 std::\_\_debug::bitset< \_Nb > Class Template Reference**

```
#include <bitset>
```

**Public Types**

- typedef \_Base::reference **reference**

**Public Member Functions**

- constexpr **bitset** (const \_Base &\_\_x)
- template<typename \_CharT, typename = \_Require<is\_trivially\_copyable<\_CharT>, is\_standard\_layout<\_CharT>, is\_trivially\_default\_constructible<\_CharT>, \_not\_<is\_array<\_CharT>>>>
 constexpr **bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>
 constexpr **bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc>
 constexpr **bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::npos))
- constexpr **bitset** (unsigned long long \_\_val) noexcept
- constexpr const \_Base & **\_M\_base** () const noexcept
- constexpr \_Base & **\_M\_base** () noexcept
- constexpr **bitset**< \_Nb > & **flip** () noexcept
- constexpr **bitset**< \_Nb > & **flip** (size\_t \_\_pos)
- constexpr **bitset**< \_Nb > & **operator&=** (const **bitset**< \_Nb > &\_\_rhs) noexcept
- constexpr **bitset**< \_Nb > **operator<<** (size\_t \_\_pos) const noexcept
- constexpr **bitset**< \_Nb > & **operator<=** (size\_t \_\_pos) noexcept
- constexpr bool **operator==** (const **bitset**< \_Nb > &\_\_rhs) const noexcept
- constexpr **bitset**< \_Nb > **operator>>** (size\_t \_\_pos) const noexcept
- constexpr **bitset**< \_Nb > & **operator>=** (size\_t \_\_pos) noexcept
- constexpr reference **operator[]** (size\_t \_\_pos)
- constexpr bool **operator[]** (size\_t \_\_pos) const
- constexpr **bitset**< \_Nb > & **operator^=** (const **bitset**< \_Nb > &\_\_rhs) noexcept
- constexpr **bitset**< \_Nb > & **operator|=** (const **bitset**< \_Nb > &\_\_rhs) noexcept
- constexpr **bitset**< \_Nb > **operator~** () const noexcept
- constexpr **bitset**< \_Nb > & **reset** () noexcept
- constexpr **bitset**< \_Nb > & **reset** (size\_t \_\_pos)
- constexpr **bitset**< \_Nb > & **set** () noexcept

- constexpr `bitset< _Nb > & set (size_t __pos, bool __val=true)`
- constexpr `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_string ()` const
- template<typename `_CharT`>  
constexpr `std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > to_string ()` const
- template<typename `_CharT`, typename `_Traits`>  
constexpr `std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string ()` const
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`>  
constexpr `std::basic_string< _CharT, _Traits, _Alloc > to_string ()` const
- template<class `_CharT`>  
constexpr `std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > to_string (`  $\leftrightarrow$  `_CharT __zero, _CharT __one=_CharT('1'))` const
- template<class `_CharT`, class `_Traits`>  
constexpr `std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string (` `_CharT __zero, _CharT` `__one=_CharT('1'))` const
- template<class `_CharT`, class `_Traits`, class `_Alloc`>  
constexpr `std::basic_string< _CharT, _Traits, _Alloc > to_string (` `_CharT __zero, _CharT __one=_CharT('1'))` const
- constexpr `std::basic_string< char, std::char_traits< char >, std::allocator< char > > to_string (char __zero, char __one='1')` const

### 6.277.1 Detailed Description

template<size\_t `_Nb`>

class `std::__debug::bitset< _Nb >`

Class `std::bitset` with additional safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 6.278 `std::bitset< _Nb >` Class Template Reference

```
#include <bitset>
```

Inheritance diagram for `std::bitset< _Nb >`:



### Classes

- class [reference](#)

## Public Member Functions

- constexpr [bitset](#) () noexcept
- template<typename \_CharT, typename = \_Require<is\_trivially\_copyable<\_CharT>, is\_standard\_layout<\_CharT>, is\_trivially\_default\_constructible<\_CharT>, \_not\_<is\_array<\_CharT>>>>  
constexpr [bitset](#) (const \_CharT \* \_\_str, typename \_\_bitset::\_\_string< \_CharT >::size\_type \_\_n= \_\_bitset::\_\_string< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > & \_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > & \_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > & \_\_s, size\_t \_\_position=0)
- constexpr [bitset](#) (unsigned long long \_\_val) noexcept
- constexpr size\_t [Find\\_first](#) () const noexcept
- constexpr size\_t [Find\\_next](#) (size\_t \_\_prev) const noexcept
- constexpr bool [all](#) () const noexcept
- constexpr bool [any](#) () const noexcept
- constexpr size\_t [count](#) () const noexcept
- constexpr [bitset](#)< \_Nb > & [flip](#) () noexcept
- constexpr [bitset](#)< \_Nb > & [flip](#) (size\_t \_\_position)
- constexpr bool [none](#) () const noexcept
- constexpr [bitset](#)< \_Nb > [operator~](#) () const noexcept
- constexpr [bitset](#)< \_Nb > & [reset](#) () noexcept
- constexpr [bitset](#)< \_Nb > & [reset](#) (size\_t \_\_position)
- constexpr [bitset](#)< \_Nb > & [set](#) () noexcept
- constexpr [bitset](#)< \_Nb > & [set](#) (size\_t \_\_position, bool \_\_val=true)
- constexpr size\_t [size](#) () const noexcept
- constexpr bool [test](#) (size\_t \_\_position) const
- constexpr [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) () const
- template<class \_CharT>  
constexpr [std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) () const
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) () const
- template<class \_CharT>  
constexpr [std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<class \_CharT, class \_Traits, class \_Alloc>  
constexpr [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > [to\\_string](#) (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- constexpr [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > [to\\_string](#) (char \_\_zero, char \_\_one='1') const
- constexpr unsigned long long [to\\_ullong](#) () const
- constexpr unsigned long [to\\_ulong](#) () const
- constexpr [bitset](#)< \_Nb > & [operator&=](#) (const [bitset](#)< \_Nb > & \_\_rhs) noexcept

- constexpr `bitset<_Nb> & operator|=` (const `bitset<_Nb> &__rhs`) noexcept
- constexpr `bitset<_Nb> & operator^=` (const `bitset<_Nb> &__rhs`) noexcept
- constexpr `bitset<_Nb> & operator<<=` (size\_t \_\_position) noexcept
- constexpr `bitset<_Nb> & operator>>=` (size\_t \_\_position) noexcept
- constexpr `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos) noexcept
- constexpr `bitset<_Nb> & _Unchecked_set` (size\_t \_\_pos, int \_\_val) noexcept
- constexpr `bitset<_Nb> & _Unchecked_reset` (size\_t \_\_pos) noexcept
- constexpr `bitset<_Nb> & _Unchecked_flip` (size\_t \_\_pos) noexcept
- constexpr bool `_Unchecked_test` (size\_t \_\_pos) const noexcept
- constexpr `reference operator[]` (size\_t \_\_position)
- constexpr bool `operator[]` (size\_t \_\_position) const
- constexpr bool `operator==` (const `bitset<_Nb> &__rhs`) const noexcept
- constexpr `bitset<_Nb> operator<<` (size\_t \_\_position) const noexcept
- constexpr `bitset<_Nb> operator>>` (size\_t \_\_position) const noexcept

## Friends

- template<class \_CharT, class \_Traits, size\_t \_Nb2>  
`std::basic_ostream<_CharT, _Traits> & operator<<` (`std::basic_ostream<_CharT, _Traits> &`, const `bitset<_Nb2> &`)
- template<class \_CharT, class \_Traits, size\_t \_Nb2>  
`std::basic_istream<_CharT, _Traits> & operator>>` (`std::basic_istream<_CharT, _Traits> &`, `bitset<_Nb2> &`)
- class **reference**
- struct `std::hash<bitset>`

## 6.278.1 Detailed Description

**template<size\_t \_Nb>**  
**class std::bitset<\_Nb>**

The `bitset` class represents a *fixed-size* sequence of bits.

(Note that `bitset` does *not* meet the formal requirements of a `container`. Mainly, it lacks iterators.)

The template argument, `Nb`, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let `B` be the number of bits in a word, then  $(Nb + (B - 1)) / B$  words will be used for storage. `B - Nb % B` bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as a *simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index `Nb-1` in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
 long a = 'a';
 bitset<10> b(a);

 cout << "b('a') is " << b << endl;

 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_containers.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html) for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the bitset is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

## 6.278.2 Constructor & Destructor Documentation

### bitset() [1/5]

```
template<size_t _Nb>
std::bitset<_Nb>::bitset () [inline], [constexpr], [noexcept]
```

All bits set to zero.

Referenced by `operator&=()`, `operator<<()`, `operator<<=()`, `operator==()`, `operator>>()`, `operator^=()`, `operator|=()`, and `operator~()`.

### bitset() [2/5]

```
template<size_t _Nb>
std::bitset<_Nb>::bitset (
 unsigned long long __val) [inline], [constexpr], [noexcept]
```

Initial bits bitwise-copied from a single word (others set to zero).

### bitset() [3/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset<_Nb>::bitset (
 const std::basic_string<_CharT, _Traits, _Alloc> & __s,
 size_t __position = 0) [inline], [explicit], [constexpr]
```

Use a subset of a string.

### Parameters

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__s</code>        | A string of 0 and 1 characters.                                            |
| <code>__position</code> | Index of the first character in <code>__s</code> to use; defaults to zero. |

**Exceptions**

|                                                    |                                                                |
|----------------------------------------------------|----------------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a>     | If <code>__position &gt; __s.size()</code> .                   |
| <a href="#"><code>std::invalid_argument</code></a> | If a character appears in the string which is neither 0 nor 1. |

References [`std::basic\_string<\_CharT, \_Traits, \_Alloc>::npos`](#).

**bitset() [4/5]**

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset<_Nb>::bitset (
 const std::basic_string<_CharT, _Traits, _Alloc> & __s,
 size_t __position,
 size_t __n) [inline], [constexpr]
```

Use a subset of a string.

**Parameters**

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__s</code>        | A string of 0 and 1 characters.                          |
| <code>__position</code> | Index of the first character in <code>__s</code> to use. |
| <code>__n</code>        | The number of characters to copy.                        |

**Exceptions**

|                                                    |                                                                |
|----------------------------------------------------|----------------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a>     | If <code>__position &gt; __s.size()</code> .                   |
| <a href="#"><code>std::invalid_argument</code></a> | If a character appears in the string which is neither 0 nor 1. |

**bitset() [5/5]**

```
template<size_t _Nb>
template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_↵
layout<_CharT>, is_trivially_default_constructible<_CharT>, __not_<is_array<_CharT>>>>
std::bitset<_Nb>::bitset (
 const _CharT * __str,
 typename __bitset<_Nb>::__string<_CharT>::size_type __n = __bitset<_Nb>::__↵
string<_CharT>::npos,
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1')) [inline], [explicit], [constexpr]
```

Construct from a character array.

**Parameters**

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>__str</code> | An array of characters <code>__zero</code> and <code>__one</code> . |
| <code>__n</code>   | The number of characters to use.                                    |

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__zero</code> | The character corresponding to the value 0. |
| <code>__one</code>  | The character corresponding to the value 1. |

**Exceptions**

|                                    |                                                                                                    |
|------------------------------------|----------------------------------------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither <code>__zero</code> nor <code>__one</code> . |
|------------------------------------|----------------------------------------------------------------------------------------------------|

References [std::basic\\_string<\\_CharT>::npos](#).

**6.278.3 Member Function Documentation****all()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::all () const [inline], [constexpr], [noexcept]
```

Tests whether all the bits are on.

**Returns**

True if all the bits are set.

**any()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::any () const [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

**count()**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::count () const [inline], [constexpr], [noexcept]
```

Returns the number of bits which are set.

**flip() [1/2]**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::flip () [inline], [constexpr], [noexcept]
```

Toggles every bit to its opposite value.

**flip() [2/2]**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::flip (
 size_t __position) [inline], [constexpr]
```

Toggles a given bit to its opposite value.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

References [\\_Unchecked\\_flip\(\)](#).

**none()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::none () const [inline], [constexpr], [noexcept]
Tests whether any of the bits are on.
```

**Returns**

True if none of the bits are set.

**operator&=()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator&= (
 const bitset<_Nb> & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

References [bitset\(\)](#).

**operator<<()**

```
template<size_t _Nb>
bitset<_Nb> > std::bitset<_Nb>::operator<< (
 size_t __position) const [inline], [constexpr], [noexcept]
```

Self-explanatory.

References [bitset\(\)](#).

**operator<<=()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator<<= (
 size_t __position) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|



These should be self-explanatory.

References [bitset\(\)](#), and [operator<=<\(\)](#).

Referenced by [operator<=<\(\)](#).

### operator==( )

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator== (
 const bitset< _Nb > & __rhs) const [inline], [constexpr], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

References [bitset\(\)](#).

### operator>>( )

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator>> (
 size_t __position) const [inline], [constexpr], [noexcept]
```

Self-explanatory.

References [bitset\(\)](#).

### operator>>=( )

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator>>= (
 size_t __position) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

#### Parameters

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|

These should be self-explanatory.

### operator[] ( ) [1/2]

```
template<size_t _Nb>
reference std::bitset< _Nb >::operator[] (
 size_t __position) [inline], [constexpr]
```

Array-indexing support.

#### Parameters

|                         |                                         |
|-------------------------|-----------------------------------------|
| <code>__position</code> | Index into the <a href="#">bitset</a> . |
|-------------------------|-----------------------------------------|

#### Returns

A bool for a *const* [bitset](#). For non-const bitsets, an instance of the reference proxy class.

#### Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

**operator[]()** [2/2]

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator[] (
 size_t __position) const [inline], [constexpr]
```

Array-indexing support.

**Parameters**

|                         |                        |
|-------------------------|------------------------|
| <code>__position</code> | Index into the bitset. |
|-------------------------|------------------------|

**Returns**

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme  
References [\\_Unchecked\\_test\(\)](#).

**operator^=()**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator^= (
 const bitset< _Nb > & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

References [bitset\(\)](#).

**operator" |=()**

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator|= (
 const bitset< _Nb > & __rhs) [inline], [constexpr], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

References [bitset\(\)](#).

**operator~()**

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::operator~ () const [inline], [constexpr], [noexcept]
```

See the no-argument `flip()`.

References [bitset\(\)](#).

**reset()** [1/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::reset () [inline], [constexpr], [noexcept]
```

Sets every bit to false.

**reset()** [2/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::reset (
 size_t __position) [inline], [constexpr]
```

Sets a given bit to false.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

**Exceptions**

|                                   |                                              |
|-----------------------------------|----------------------------------------------|
| <a href="#">std::out_of_range</a> | If <i>pos</i> is bigger the size of the set. |
|-----------------------------------|----------------------------------------------|

Same as writing `set (pos, false)`.

References [\\_Unchecked\\_reset\(\)](#).

**set()** [1/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::set () [inline], [constexpr], [noexcept]
```

Sets every bit to true.

**set()** [2/2]

```
template<size_t _Nb>
bitset<_Nb> & std::bitset<_Nb>::set (
 size_t __position,
 bool __val = true) [inline], [constexpr]
```

Sets a given bit to a particular value.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <code>__position</code> | The index of the bit.                   |
| <code>__val</code>      | Either true or false, defaults to true. |

**Exceptions**

|                                                |                                              |
|------------------------------------------------|----------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos</i> is bigger the size of the set. |
|------------------------------------------------|----------------------------------------------|

References [\\_Unchecked\\_set\(\)](#).

**size()**

```
template<size_t _Nb>
size_t std::bitset< _Nb >::size () const [inline], [constexpr], [noexcept]
```

Returns the total number of bits.

**test()**

```
template<size_t _Nb>
bool std::bitset< _Nb >::test (
 size_t __position) const [inline], [constexpr]
```

Tests the value of a bit.

**Parameters**

|                         |                     |
|-------------------------|---------------------|
| <code>__position</code> | The index of a bit. |
|-------------------------|---------------------|

**Returns**

The value at *pos*.

**Exceptions**

|                                                |                                              |
|------------------------------------------------|----------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <i>pos</i> is bigger the size of the set. |
|------------------------------------------------|----------------------------------------------|

References [\\_Unchecked\\_test\(\)](#).

**to\_string()**

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::basic_string< _CharT, _Traits, _Alloc > std::bitset< _Nb >::to_string () const [inline],
[constexpr]
```

Returns a character interpretation of the bitset.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

**to\_ulong()**

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong () const [inline], [constexpr]
```

Returns a numerical interpretation of the bitset.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                     |                                                                   |
|-------------------------------------|-------------------------------------------------------------------|
| <a href="#">std::overflow_error</a> | If there are too many bits to be represented in an unsigned long. |
|-------------------------------------|-------------------------------------------------------------------|

The documentation for this class was generated from the following file:

- [bitset](#)

**6.279 std::tr2::bool\_set Class Reference**

```
#include <bool_set>
```

**Public Member Functions**

- constexpr [bool\\_set](#) ()
- constexpr [bool\\_set](#) (bool \_\_t)
- bool **contains** ([bool\\_set](#) \_\_b) const
- bool **equals** ([bool\\_set](#) \_\_b) const
- bool **is\_emptyset** () const
- bool **is\_indeterminate** () const
- bool **is\_singleton** () const
- **operator bool** () const

**Static Public Member Functions**

- static [bool\\_set](#) **emptyset** ()
- static [bool\\_set](#) **indeterminate** ()

**Friends**

- [bool\\_set](#) **operator!** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT, typename Traits>  
[std::basic\\_ostream](#)< CharT, Traits > & **operator<<** ([std::basic\\_ostream](#)< CharT, Traits > &\_\_out, [bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator==** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT, typename Traits>  
[std::basic\\_istream](#)< CharT, Traits > & **operator>>** ([std::basic\\_istream](#)< CharT, Traits > &\_\_in, [bool\\_set](#) &\_\_b)
- [bool\\_set](#) **operator^** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)

**6.279.1 Detailed Description**

[bool\\_set](#)

See N2136, Bool\_set: multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

### 6.279.2 Constructor & Destructor Documentation

#### **bool\_set()** [1/2]

```
std::tr2::bool_set::bool_set () [inline], [constexpr]
```

Default constructor.

Referenced by [equals\(\)](#).

#### **bool\_set()** [2/2]

```
std::tr2::bool_set::bool_set (
 bool __t) [inline], [constexpr]
```

Constructor from bool.

### 6.279.3 Member Function Documentation

#### **equals()**

```
bool std::tr2::bool_set::equals (
 bool_set __b) const [inline]
```

Return true if states are equal.

References [bool\\_set\(\)](#).

#### **is\_emptyset()**

```
bool std::tr2::bool_set::is_emptyset () const [inline]
```

Return true if this is empty.

#### **is\_indeterminate()**

```
bool std::tr2::bool_set::is_indeterminate () const [inline]
```

Return true if this is indeterminate.

#### **is\_singleton()**

```
bool std::tr2::bool_set::is_singleton () const [inline]
```

Return true if this is false or true (normal boolean).

Referenced by [operator bool\(\)](#).

#### **operator bool()**

```
std::tr2::bool_set::operator bool () const [inline]
```

Conversion to bool.

References [is\\_singleton\(\)](#).

The documentation for this class was generated from the following files:

- [bool\\_set](#)
- [bool\\_set.tcc](#)

### 6.280 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >` Struct Template Reference

```
#include <branch_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`:



### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `rebind_k::const_reference` **key\_const\_reference**
- typedef `value_type::first_type` **key\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `rebind_traits< _Alloc, rckey_type >` **rebind\_k**
- typedef `rebind_traits< _Alloc, rcvalue_type >` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

### Protected Member Functions

- virtual `it_type end ()=0`
- `it_type end_iterator () const`

### Static Protected Member Functions

- static `key_const_reference extract_key (const_reference r_val)`

#### 6.280.1 Detailed Description

**template<typename Node\_Cltr, typename Node\_Itr, typename \_Alloc>**  
**struct** `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`

Primary template, base class for branch structure policies.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 6.281 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

```
#include <branch_policy.hpp>
```

### Protected Types

- typedef rebind\_v::const\_pointer **const\_pointer**
- typedef rebind\_v::const\_reference **const\_reference**
- typedef Node\_Cltr::value\_type **it\_type**
- typedef rebind\_v::const\_reference **key\_const\_reference**
- typedef value\_type **key\_type**
- typedef remove\_const< key\_type >::type **rckey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef [rebind\\_traits](#)< \_Alloc, rckey\_type > **rebind\_k**
- typedef [rebind\\_traits](#)< \_Alloc, rcvalue\_type > **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef [std::iterator\\_traits](#)< it\_type >::value\_type **value\_type**

### Protected Member Functions

- virtual it\_type **end** () const =0
- it\_type **end\_iterator** () const

### Static Protected Member Functions

- static key\_const\_reference **extract\_key** (const\_reference r\_val)

#### 6.281.1 Detailed Description

```
template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >
```

Specialization for const iterators.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

## 6.282 std::cauchy\_distribution<\_RealType> Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a, \_RealType \_\_b=1.0)
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)



- template<typename \_UniformRandomNumberGenerator>  
void **generate** (result\_type \*\_\_f, result\_type \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- result\_type **max** () const
- result\_type **min** () const
- template<typename \_UniformRandomNumberGenerator>  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
result\_type **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const param\_type &\_\_p)
- param\_type **param** () const
- void **param** (const param\_type &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const cauchy\_distribution &\_\_d1, const cauchy\_distribution &\_\_d2)

### 6.282.1 Detailed Description

template<typename \_RealType = double>  
class std::cauchy\_distribution<\_RealType>

A cauchy\_distribution random number distribution.

The formula for the normal probability mass function is  $p(x|a, b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Since

C++11

### 6.282.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution<_RealType>::result_type
```

The type of the range of the distribution.

### 6.282.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution<_RealType>::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution<_RealType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::cauchy_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param()** [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param()** [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**6.282.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
 const cauchy_distribution< _RealType > & __d1,
 const cauchy_distribution< _RealType > & __d2) [friend]
```

Return true if two Cauchy distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.283 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference**

```
#include <hash_policy.hpp>
```

## Public Types

- enum { [external\\_load\\_access](#) }
- typedef `Size_Type` **size\_type**

## Public Member Functions

- [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#) (float load=0.5)
- float [get\\_load](#) () const
- void [set\\_load](#) (float load)
- void **swap** ([cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)< `External_Load_Access`, `Size_Type` > &other)

## Protected Member Functions

- bool [is\\_grow\\_needed](#) (size\_type size, size\_type num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) (size\_type num\_entries)
- void [notify\\_externally\\_resized](#) (size\_type new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) (size\_type num\_entries)
- void [notify\\_resized](#) (size\_type new\_size)

### 6.283.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

### 6.283.2 Member Enumeration Documentation

#### anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

#### Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 6.283.3 Constructor & Destructor Documentation

#### **cc\_hash\_max\_collision\_check\_resize\_trigger()**

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_↵
hash_max_collision_check_resize_trigger (
 float load = 0.5)
```

Default constructor, or constructor taking load, a \_\_load factor which it will attempt to maintain.

### 6.283.4 Member Function Documentation

#### **get\_load()**

```
template<bool External_Load_Access, typename Size_Type>
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::get_load () const [inline]
```

Returns the current load.

#### **is\_grow\_needed()**

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::is_grow_needed (
 size_type size,
 size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

#### **is\_resize\_needed()**

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

#### **notify\_cleared()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_cleared () [protected]
```

Notifies the table was cleared.

References [notify\\_resized\(\)](#).

#### **notify\_erase\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### **notify\_erase\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_erase\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_start () [inline], [protected]
```

Notifies an erase search started.

**notify\_erased()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erased (
 size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

**notify\_externally\_resized()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_externally_resized (
 size_type new_size) [protected]
```

Notifies the table was resized externally.

**notify\_find\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_find\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_start () [inline], [protected]
```

Notifies a find search started.

**notify\_insert\_search\_collision()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

#### **notify\_insert\_search\_start()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_start () [inline], [protected]
```

Notifies an insert search started.

#### **notify\_inserted()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_inserted (
 size_type num_entries) [inline], [protected]
```

Notifies an element was inserted.

#### **notify\_resized()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Referenced by [notify\\_cleared\(\)](#).

#### **set\_load()**

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::set_load (
 float load)
```

Sets the load; does not resize the container.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

### **6.284 \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc > Class Template Reference**

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy,`

Store\_Hash, \_Alloc >:



## Public Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Resize\_Policy **resize\_policy**

## Public Member Functions

- [cc\\_hash\\_table](#) ()
- **cc\_hash\_table** (const [cc\\_hash\\_table](#) &other)
- [cc\\_hash\\_table](#) (const hash\_fn &h)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- [cc\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It>  
[cc\\_hash\\_table](#) (It first, It last)
- template<typename It>  
[cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h)
- template<typename It>  
[cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It>  
[cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It>  
[cc\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- [cc\\_hash\\_table](#) & **operator=** (const [cc\\_hash\\_table](#) &other)
- void **swap** ([cc\\_hash\\_table](#) &other)

### 6.284.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A collision-chaining hash-based associative container.

#### Template Parameters

|                      |                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                       |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                       |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                  |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                   |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true                                         |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                 |

Base tag choices are: cc\_hash\_tag.

Base is basic\_hash\_table.

### 6.284.2 Constructor & Destructor Documentation

#### cc\_hash\_table() [1/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table () [inline]
Default constructor.
```

#### cc\_hash\_table() [2/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (
 const hash_fn & h) [inline]
Constructor taking some policy objects. r_hash_fn will be copied by the Hash_Fn object of the container object.
```



**cc\_hash\_table()** [3/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

**cc\_hash\_table()** [4/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

**cc\_hash\_table()** [5/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch,
 const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

**cc\_hash\_table()** [6/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
template<typename It>
```

```
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

#### **cc\_hash\_table()** [7/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

#### **cc\_hash\_table()** [8/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

#### **cc\_hash\_table()** [9/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
```

```
const eq_fn & e,
const comb_hash_fn & ch) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, and r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object.

### cc\_hash\_table() [10/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename _Alloc = std::allocator<char>>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch,
 const resize_policy & rp) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object. The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 6.285 \_\_gnu\_pbds::cc\_hash\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



#### 6.285.1 Detailed Description

Collision-chaining hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 6.286 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

```
#include <cc_ht_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb`↵

`_Hash_Fn, Resize_Policy >`:



## Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **cc\_ht\_map** (const cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)
- **cc\_ht\_map** (const Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &, const Resize\_Policy &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Hash\_Fn & **get\_comb\_hash\_fn** ()
- const Comb\_Hash\_Fn & **get\_comb\_hash\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Resize\_Policy & **get\_resize\_policy** ()
- const Resize\_Policy & **get\_resize\_policy** () const
- void **initialize** ()
- **std::pair**< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class **const\_iterator\_**
- class **iterator\_**

### 6.286.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn,
Resize_Policy >
```

A collision-chaining hash-based container.

#### Template Parameters

|                      |                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                               |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                               |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                              |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                            |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                         |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                   |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default <code>direct_mask_range_hashing</code> . |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                  |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

### 6.286.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
anonymous enum
Value stores hash, true or false.
```

### 6.286.3 Member Function Documentation

#### `empty()`

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_
Fn, Resize_Policy >::empty () const [inline], [nodiscard]
True if size() == 0.
```

#### `get_comb_hash_fn()` [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()
Return current comb_hash_fn.
```

**get\_comb\_hash\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const
Return current const comb_hash_fn.
```

**get\_eq\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_eq_fn ()
Return current eq_fn.
```

**get\_eq\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const
Return current const eq_fn.
```

**get\_hash\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_hash_fn ()
Return current hash_fn.
```

**get\_hash\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const
Return current const hash_fn.
```

**get\_resize\_policy()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()
Return current resize_policy.
```

**get\_resize\_policy()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const
Return current const resize_policy.
```

The documentation for this class was generated from the following file:



- [cc\\_ht\\_map.hpp](#)

## 6.287 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

```
#include <char_traits.h>
```

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



### Public Types

- typedef `_CharT` **char\_type**
- using **comparison\_category**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)

- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

### 6.287.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::char_traits<_CharT >
```

Base class used to implement std::char\_traits.

#### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the int\_type and state\_type typedefs, and the eof() member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace std may not be specialized for fundamental types, but classes in namespace \_\_gnu\_cxx may be.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod\_char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 6.288 std::char\_traits<\_CharT > Struct Template Reference

```
#include <char_traits.h>
```

Inheritance diagram for std::char\_traits<\_CharT >:



### Public Types

- typedef \_CharT **char\_type**

- using **comparison\_category**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \*\_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 6.288.1 Detailed Description

**template<typename \_CharT>**  
**struct std::char\_traits<\_CharT>**

Basis for explicit traits specializations.

#### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

### 6.289 `std::char_traits<__gnu_cxx::character<_Value, _Int, _St>>` Struct Template Reference

```
#include <pod_char_traits.h>
```

Inheritance diagram for `std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >`:



### Public Types

- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char\_type**
- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char\_type**
- using **comparison\_category**
- using **comparison\_category**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::int_type` **int\_type**
- typedef `char_type::int_type` **int\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::off_type` **off\_type**
- typedef **streamoff** **off\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::pos_type` **pos\_type**
- typedef `fpos< state_type >` **pos\_type**
- typedef `_Char_types< __gnu_cxx::character< _Value, _Int, _St > >::state_type` **state\_type**
- typedef `char_type::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)

- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static char\_type **to\_char\_type** (const int\_type &\_\_i)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static int\_type **to\_int\_type** (const char\_type &\_\_c)

### 6.289.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St>
struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
```

char\_traits<\_\_gnu\_cxx::character> specialization.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 6.290 std::char\_traits< char > Struct Reference

```
#include <char_traits.h>
```

Inheritance diagram for `std::char_traits< char >`:



### Public Types

- typedef char **char\_type**
- typedef char **char\_type**
- using **comparison\_category**
- using **comparison\_category**
- typedef int **int\_type**
- typedef `_Char_types< char >::int_type` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `_Char_types< char >::off_type` **off\_type**
- typedef `streampos` **pos\_type**
- typedef `_Char_types< char >::pos_type` **pos\_type**
- typedef `mbstate_t` **state\_type**
- typedef `_Char_types< char >::state_type` **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept

- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr size\_t **length** (const char\_type \* \_\_s)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

### 6.290.1 Detailed Description

#### 21.1.3.1 char\_traits specializations

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 6.291 std::char\_traits< wchar\_t > Struct Reference

```
#include <char_traits.h>
```

Inheritance diagram for std::char\_traits< wchar\_t >:



## Public Types

- typedef wchar\_t **char\_type**
- typedef wchar\_t **char\_type**
- using **comparison\_category**
- using **comparison\_category**
- typedef wint\_t **int\_type**
- typedef \_Char\_types< wchar\_t >::int\_type **int\_type**
- typedef **streamoff** **off\_type**
- typedef \_Char\_types< wchar\_t >::off\_type **off\_type**
- typedef **wstreampos** **pos\_type**
- typedef \_Char\_types< wchar\_t >::pos\_type **pos\_type**
- typedef mbstate\_t **state\_type**
- typedef \_Char\_types< wchar\_t >::state\_type **state\_type**

## Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, size\_t \_\_n, char\_type \_\_a)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr size\_t **length** (const char\_type \* \_\_s)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept



- static constexpr `char_type` **to\_char\_type** (const `int_type` &\_\_c)
- static constexpr `char_type` **to\_char\_type** (const `int_type` &\_\_c)
- static constexpr `char_type` **to\_char\_type** (const `int_type` &\_\_c) noexcept
- static constexpr `int_type` **to\_int\_type** (const `char_type` &\_\_c)
- static constexpr `int_type` **to\_int\_type** (const `char_type` &\_\_c)
- static constexpr `int_type` **to\_int\_type** (const `char_type` &\_\_c) noexcept

### 6.291.1 Detailed Description

21.1.3.2 `char_traits` specializations

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 6.292 `__gnu_cxx::character<_Value, _Int, _St>` Struct Template Reference

```
#include <pod_char_traits.h>
```

### Public Types

- typedef `character<_Value, _Int, _St>` **char\_type**
- typedef `_Int` **int\_type**
- typedef `_St` **state\_type**
- typedef `_Value` **value\_type**

### Static Public Member Functions

- template<typename V2>  
static `char_type` **from** (const V2 &v)
- template<typename V2>  
static V2 **to** (const `char_type` &c)

### Public Attributes

- `value_type` **value**

### 6.292.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character<_Value, _Int, _St>
```

A POD class that serves as a character abstraction class.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 6.293 `std::chi_squared_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct `param_type`

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **chi\_squared\_distribution** (`_RealType __n`)
- **chi\_squared\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- `_RealType` **n** () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_ostream`< `_CharT`, `_Traits` > & **operator<<** (`std::basic_ostream`< `_CharT`, `_Traits` > &\_\_os, const `std::chi_squared_distribution`< `_RealType1` > &\_\_x)
- bool **operator==** (const `chi_squared_distribution` &\_\_d1, const `chi_squared_distribution` &\_\_d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_istream`< `_CharT`, `_Traits` > & **operator>>** (`std::basic_istream`< `_CharT`, `_Traits` > &\_\_is, `std::chi_squared_distribution`< `_RealType1` > &\_\_x)

### 6.293.1 Detailed Description

**template<typename `_RealType` = double>**  
**class `std::chi_squared_distribution`< `_RealType` >**

A `chi_squared_distribution` random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Since

C++11

### 6.293.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.293.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::chi_squared_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

## 6.293.4 Friends And Related Symbol Documentation

### operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::chi_squared_distribution< _RealType1 > & __x) [friend]
Inserts a chi_squared_distribution random number distribution __x into the output stream __os.
```

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A chi_squared_distribution random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

### operator==

```
template<typename _RealType = double>
bool operator== (
 const chi_squared_distribution< _RealType > & __d1,
 const chi_squared_distribution< _RealType > & __d2) [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

### operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::chi_squared_distribution< _RealType1 > & __x) [friend]
Extracts a chi_squared_distribution random number distribution __x from the input stream __is.
```

#### Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__is</code> | An input stream.                                           |
| <code>__x</code>  | A chi_squared_distribution random number generator engine. |

#### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.294 std::codecvt< \_InternT, \_ExternT, \_StateT > Class Template Reference**

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< \_InternT, \_ExternT, \_StateT >:

**Public Types**

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

**Public Member Functions**

- **codecvt** (`_c_locale __cloc, size_t __refs=0`)
- **codecvt** (`size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__from_next, intern_type *__to, intern_type *__to_end, intern_type *__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- **int max\_length** () const throw ()

- result [out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result [unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual bool [do\\_always\\_noconv](#) () const throw ()
- virtual int [do\\_encoding](#) () const throw ()
- virtual result [do\\_in](#) (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int [do\\_length](#) (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int [do\\_max\\_length](#) () const throw ()
- virtual result [do\\_out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result [do\\_unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to←\_\_next) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_codecvt](#)

#### 6.294.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt< _InternT, _ExternT, _StateT >
```

Primary class template codecvt.

NB: Generic, mostly useless implementation.

#### 6.294.2 Member Function Documentation

##### [do\\_always\\_noconv\(\)](#)

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ()
[protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#).

**do\_encoding()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<_InternT, _ExternT, _StateT >::do_encoding () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<_InternT, _ExternT, _StateT >::do_length (
 state_type & ,
 const extern_type * __from,
 const extern_type * __end,
 size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_max\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<_InternT, _ExternT, _StateT >::do_max_length () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

`out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< \\_InternT, \\_ExternT, \\_StateT >](#).

**in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
```



```
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The state argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how state is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

#### Parameters

|             |                                      |
|-------------|--------------------------------------|
| __state     | Persistent conversion state data.    |
| __from      | Start of input.                      |
| __from_end  | End of input.                        |
| __from_next | Returns start of unconverted data.   |
| __to        | Start of output buffer.              |
| __to_end    | End of output buffer.                |
| __to_next   | Returns start of unused output area. |

#### Returns

codecvt\_base::result.

References [do\\_out\(\)](#).

#### unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore state to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types. The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 6.295 `std::codecvt<_InternT, _ExternT, encoding_state >` Class Template Reference

```
#include <codecvt_specializations.h>
```

Inheritance diagram for `std::codecvt<_InternT, _ExternT, encoding_state >`:



#### Public Types

- `typedef state_type::descriptor_type` **descriptor\_type**

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (state\_type &\_\_enc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Public Attributes

- static `locale::id` **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

### Static Protected Member Functions

- static `__c_locale` **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static `__c_locale` **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)

- static `__c_locale` `_S_get_c_locale` ()
- static `__c_locale` `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)

## Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

### 6.295.1 Detailed Description

template<typename `_InternT`, typename `_ExternT`>  
class `std::codecvt`< `_InternT`, `_ExternT`, `encoding_state` >

`codecvt`<`InternT`, `_ExternT`, `encoding_state`> specialization.

### 6.295.2 Member Function Documentation

#### `do_always_noconv()`

```
template<typename _InternT, typename _ExternT>
bool std::codecvt< _InternT, _ExternT, encoding_state >::do_always_noconv () const throw ()
[protected], [virtual]
```

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

#### `do_encoding()`

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_encoding () const throw () [protected],
[virtual]
```

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

#### `do_in()`

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

#### `do_length()`

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
 state_type & ,
 const extern_type * __from,
 const extern_type * __end,
 size_t __max) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

**do\_max\_length()**

```
template<typename _InternT, typename _ExternT>
int std::codecvt<_InternT, _ExternT, encoding_state >::do_max_length () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_out()**

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt<_InternT, _ExternT, encoding_state >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

`do_in` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**do\_unshift()**

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt<_InternT, _ExternT, encoding_state >::do_unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state >](#).

**in() [1/2]**

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

#### in() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

#### Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()** [1/2]

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out()** [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift()** [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore `state` to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.



The source and destination character sets are determined by the facet's locale, internal and external types. The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 6.296 std::codecvt< char, char, mbstate\_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (`__c_locale __cloc, size_t __refs=0`)
- **codecvt** (`size_t __refs=0`)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- result **in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- int **length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- int **length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const

- result [out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \* \_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const
- result [unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const
- result [unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual bool [do\\_always\\_noconv](#) () const throw ()
- virtual int [do\\_encoding](#) () const throw ()
- virtual result [do\\_in](#) (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \* \_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \* \_\_&\_\_to\_next) const
- virtual int [do\\_length](#) (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int [do\\_max\\_length](#) () const throw ()
- virtual result [do\\_out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \* \_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const
- virtual result [do\\_unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \* \_\_&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_codecvt](#)

### Friends

- class [messages< char >](#)

#### 6.296.1 Detailed Description

class codecvt<char, char, mbstate\_t> specialization.

#### 6.296.2 Member Function Documentation

##### [do\\_always\\_noconv\(\)](#)

virtual bool [std::codecvt< char, char, mbstate\\_t >::do\\_always\\_noconv](#) () const throw () [protected], [virtual]

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

**do\_encoding()**

```
virtual int std::codecvt< char, char, mbstate_t >::do_encoding () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char, char, mbstate\_t >.

**do\_in()**

```
virtual result std::codecvt< char, char, mbstate_t >::do_in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char, char, mbstate\_t >.

**do\_length()**

```
virtual int std::codecvt< char, char, mbstate_t >::do_length (
 state_type & ,
 const extern_type * __from,
 const extern_type * __end,
 size_t __max) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char, char, mbstate\_t >.

**do\_max\_length()**

```
virtual int std::codecvt< char, char, mbstate_t >::do_max_length () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char, char, mbstate\_t >.

**do\_out()**

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

**See also**

[do\\_out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char, char, mbstate\_t >.

**do\_unshift()**

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
 state_type & __state,
```

```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char, char, mbstate\\_t >](#).

## in() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

## in() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
```

```
intern_type * __to_end,
intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

`codecvt_base::result`.

### out() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the

output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

codecvt\_base::result.

### out() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |
| <code>__to</code>        | Start of output buffer.            |

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift() [1/2]**

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.



The source and destination character sets are determined by the facet's locale, internal and external types. The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

#### Returns

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 6.297 std::codecvt< char16\_t, char, mbstate\_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt< char16_t, char, mbstate_t >`:



#### Public Types

- typedef char **extern\_type**

- typedef char16\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()

- static const char \* `_S_get_c_name` () throw ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, const char \* `__s`)

### Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

#### 6.297.1 Detailed Description

Class `codecvt<char16_t, char, mbstate_t>` specialization.  
Converts between UTF-16 and UTF-8.

#### 6.297.2 Member Function Documentation

##### `do_always_noconv()`

virtual bool `std::codecvt< char16_t, char, mbstate_t >::do_always_noconv` () const throw ( ) [protected],  
[virtual]

Implements `std::__codecvt_abstract_base< char16_t, char, mbstate_t >`.

##### `do_encoding()`

virtual int `std::codecvt< char16_t, char, mbstate_t >::do_encoding` () const throw ( ) [protected],  
[virtual]

Implements `std::__codecvt_abstract_base< char16_t, char, mbstate_t >`.

##### `do_in()`

virtual result `std::codecvt< char16_t, char, mbstate_t >::do_in` (  
state\_type & `__state`,  
const extern\_type \* `__from`,  
const extern\_type \* `__from_end`,  
const extern\_type \* & `__from_next`,  
intern\_type \* `__to`,  
intern\_type \* `__to_end`,  
intern\_type \* & `__to_next`) const [protected], [virtual]

Implements `std::__codecvt_abstract_base< char16_t, char, mbstate_t >`.

##### `do_length()`

virtual int `std::codecvt< char16_t, char, mbstate_t >::do_length` (  
state\_type & ,  
const extern\_type \* `__from`,  
const extern\_type \* `__end`,  
size\_t `__max`) const [protected], [virtual]

Implements `std::__codecvt_abstract_base< char16_t, char, mbstate_t >`.

##### `do_max_length()`

virtual int `std::codecvt< char16_t, char, mbstate_t >::do_max_length` () const throw ( ) [protected],  
[virtual]

Implements `std::__codecvt_abstract_base< char16_t, char, mbstate_t >`.

**do\_out()**

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

**See also**

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**do\_unshift()**

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char16\\_t, char, mbstate\\_t >](#).

**in()** [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__state</code> | Persistent conversion state data. |
|----------------------|-----------------------------------|

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**in()** [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
```

```
extern_type * __to_end,
extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

|             |                                      |
|-------------|--------------------------------------|
| __state     | Persistent conversion state data.    |
| __from      | Start of input.                      |
| __from_end  | End of input.                        |
| __from_next | Returns start of unconverted data.   |
| __to        | Start of output buffer.              |
| __to_end    | End of output buffer.                |
| __to_next   | Returns start of unused output area. |

### Returns

codecvt\_base::result.

### unshift() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**6.298 std::codecvt< char32\_t, char, mbstate\_t > Class Reference**

```
#include <codecvt.h>
```



Inheritance diagram for std::codecvt< char32\_t, char, mbstate\_t >:



### Public Types

- typedef char **extern\_type**
- typedef char32\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual bool [do\\_always\\_noconv](#) () const throw ()
- virtual int [do\\_encoding](#) () const throw ()
- virtual result [do\\_in](#) (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int [do\\_length](#) (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int [do\\_max\\_length](#) () const throw ()
- virtual result [do\\_out](#) (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result [do\\_unshift](#) (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to←\_\_next) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_codecvt](#)

### 6.298.1 Detailed Description

Class `codecvt<char32_t, char, mbstate_t>` specialization.  
Converts between UTF-32 and UTF-8.

### 6.298.2 Member Function Documentation

#### [do\\_always\\_noconv\(\)](#)

```
virtual bool std::codecvt< char32_t, char, mbstate_t >::do_always_noconv () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

#### [do\\_encoding\(\)](#)

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_encoding () const throw () [protected],
[virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< char32\\_t, char, mbstate\\_t >](#).

**do\_in()**

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_length()**

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
 state_type & ,
 const extern_type * __from,
 const extern_type * __end,
 size_t __max) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_max\_length()**

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw () [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_out()**

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

`do_in` for more information.

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**do\_unshift()**

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

**in()** [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**in()** [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

`codecvt_base::result`.

### `out()` [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**out() [2/2]**

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**unshift() [1/2]**

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**6.299 std::codecvt< wchar\_t, char, mbstate\_t > Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:

**Public Types**

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state\_type**



**Public Member Functions**

- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

**Static Public Attributes**

- static [locale::id](#) id

**Protected Member Functions**

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

**Static Protected Member Functions**

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale __M_c_locale_codecvt`

## Friends

- class `messages< wchar_t >`

### 6.299.1 Detailed Description

Class `codecvt<wchar_t, char, mbstate_t>` specialization.  
Converts between narrow and wide characters in the native character set

### 6.299.2 Member Function Documentation

#### `do_always_noconv()`

virtual bool `std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

#### `do_encoding()`

virtual int `std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

#### `do_in()`

virtual result `std::codecvt< wchar_t, char, mbstate_t >::do_in (`  
    `state_type & __state,`  
    `const extern_type * __from,`  
    `const extern_type * __from_end,`  
    `const extern_type *& __from_next,`  
    `intern_type * __to,`  
    `intern_type * __to_end,`  
    `intern_type *& __to_next) const` [protected], [virtual]

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

#### `do_length()`

virtual int `std::codecvt< wchar_t, char, mbstate_t >::do_length (`  
    `state_type & ,`  
    `const extern_type * __from,`  
    `const extern_type * __end,`  
    `size_t __max) const` [protected], [virtual]

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

#### `do_max_length()`

virtual int `std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( )` [protected], [virtual]

Implements `std::__codecvt_abstract_base< wchar_t, char, mbstate_t >`.

**do\_out()**

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

**See also**

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**do\_unshift()**

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

**in()** [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__state</code> | Persistent conversion state data. |
|----------------------|-----------------------------------|

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

## in() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

**out()** [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
```

```
extern_type * __to_end,
extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

`codecvt_base::result`.

### unshift() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore `state` to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

**unshift() [2/2]**

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**6.300 std::codecvt\_base Class Reference**

```
#include <codecvt.h>
```

Inheritance diagram for `std::codecvt_base`:



## Public Types

- enum **result** { **ok** , **partial** , **error** , **noconv** }

### 6.300.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 6.301 `std::codecvt_byname<_InternT, _ExternT, _StateT>` Class Template Reference

```
#include <codecvt.h>
```



Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

### Public Member Functions

- **codecvt\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **codecvt\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `bool do_always_noconv ()` `const throw ()`
- virtual `int do_encoding ()` `const throw ()`
- virtual `result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type * &__from_next, intern_type *__to, intern_type *__to_end, intern_type * &__to_next)` `const`
- virtual `int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max)` `const`
- virtual `int do_max_length ()` `const throw ()`
- virtual `result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type * &__from_next, extern_type *__to, extern_type *__to_end, extern_type * &__to_next)` `const`
- virtual `result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type * &__to←__next)` `const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc)` `throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `const char * _S_get_c_name ()` `throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

## Protected Attributes

- `__c_locale _M_c_locale_codecvt`

### 6.301.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class `codecvt_byname` [22.2.1.6].

### 6.301.2 Member Function Documentation

#### `do_always_noconv()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ()
[protected], [virtual], [inherited]
Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.
```

#### `do_encoding()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw () [protected],
[virtual], [inherited]
Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.
```

**do\_in()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type * __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type * __to_next) const [protected], [virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<_InternT, _ExternT, _StateT >::do_length (
 state_type & ,
 const extern_type * __from,
 const extern_type * __end,
 size_t __max) const [protected], [virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_max\_length()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt<_InternT, _ExternT, _StateT >::do_max_length () const throw () [protected],
[virtual], [inherited]
```

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_out()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type * __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type * __to_next) const [protected], [virtual], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

`do_out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT >](#).

**do\_unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt<_InternT, _ExternT, _StateT >::do_unshift (
 state_type & __state,
 extern_type * __to,
```

```
extern_type * __to_end,
extern_type *& __to_next) const [protected], [virtual], [inherited]
```

Implements `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >`.

## in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

## out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
```

```
extern_type * __to,
extern_type * __to_end,
extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

### Returns

codecvt\_base::result.

References [do\\_out\(\)](#).

### unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

**6.302 `std::__cxx11::collate<_CharT>` Class Template Reference**

```
#include <locale_classes.h>
```

Inheritance diagram for `std::__cxx11::collate<_CharT>`:

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

**Public Member Functions**

- [collate](#) (`_c_locale __cloc, size_t __refs=0`)
- [collate](#) (`size_t __refs=0`)
- `int` [\\_M\\_compare](#) (`const _CharT *, const _CharT *`) `const throw ()`
- `size_t` [\\_M\\_transform](#) (`_CharT *, const _CharT *, size_t`) `const throw ()`
- `int` [compare](#) (`const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2`) `const`
- `long` [hash](#) (`const _CharT * __lo, const _CharT * __hi`) `const`
- [string\\_type transform](#) (`const _CharT * __lo, const _CharT * __hi`) `const`

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~collate](#) ()
- virtual int [do\\_compare](#) (const \_CharT \*\_\_lo1, const \_CharT \*\_\_hi1, const \_CharT \*\_\_lo2, const \_CharT \*\_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \*\_\_lo, const \_CharT \*\_\_hi) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_collate](#)

#### 6.302.1 Detailed Description

```
template<typename _CharT>
class std::__cxx11::collate<_CharT>
```

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

#### 6.302.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate<_CharT>::char_type
```

Public typedefs.

##### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate<_CharT>::string_type
```

Public typedefs.

#### 6.302.3 Constructor & Destructor Documentation

##### collate() [1/2]

```
template<typename _CharT>
std::__cxx11::collate<_CharT>::collate (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.  
This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).  
Referenced by [do\\_compare\(\)](#), [do\\_hash\(\)](#), and [do\\_transform\(\)](#).

#### collate() [2/2]

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
 __c_locale __cloc,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.  
This is a constructor for use by the library itself to set up new locales.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__refs</code> | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

#### ~collate()

```
template<typename _CharT>
virtual std::__cxx11::collate< _CharT >::~~collate () [inline], [protected], [virtual]
```

Destructor.

### 6.302.4 Member Function Documentation

#### compare()

```
template<typename _CharT>
int std::__cxx11::collate< _CharT >::compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [inline]
```

Compare two strings.  
This function compares two strings and returns the result by calling `collate::do_compare()`.

#### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |



**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

References [do\\_compare\(\)](#).

**do\_compare()**

```
template<typename _CharT>
int std::collate<_CharT>::do_compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [protected], [virtual]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

**See also**

[compare\(\)](#).

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

**Returns**

1 if string1 > string2, -1 if string1 < string2, else 0.

References [collate\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::length\(\)](#).

Referenced by [compare\(\)](#).

**do\_hash()**

```
template<typename _CharT>
long std::collate<_CharT>::do_hash (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Hash value.

References [collate\(\)](#).

Referenced by [hash\(\)](#).

**do\_transform()**

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

**Parameters**

|                                                                                                               |        |
|---------------------------------------------------------------------------------------------------------------|--------|
| <a href="#"> <i>__lo</i></a> | Start. |
| <a href="#"> <i>__hi</i></a> | End.   |

**Returns**

transformed string.

References [collate\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::append\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::clear\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::data\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::push\\_back\(\)](#) and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::push\\_back\(\)](#).

Referenced by [transform\(\)](#).

**hash()**

```
template<typename _CharT>
long std::__cxx11::collate< _CharT >::hash (
 const _CharT * __lo,
 const _CharT * __hi) const [inline]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do\\_hash\(\)](#).

**Parameters**

|                                                                                                                 |                  |
|-----------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#"> <i>__lo</i></a> | Start of string. |
| <a href="#"> <i>__hi</i></a> | End of string.   |

**Returns**

Hash value.

References [do\\_hash\(\)](#).

**transform()**

```
template<typename _CharT>
string_type std::__cxx11::collate<_CharT>::transform (
 const _CharT * __lo,
 const _CharT * __hi) const [inline]
```

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

**Parameters**

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

**Returns**

Transformed string\_type.

References [do\\_transform\(\)](#).

**6.302.5 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::__cxx11::collate<_CharT>::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

**6.303 std::\_\_cxx11::collate\_byname<\_CharT> Class Template Reference**

```
#include <locale_classes.h>
```

**Public Types**

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)<`_CharT`> [string\\_type](#)

**Public Member Functions**

- **collate\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **collate\_byname** (const [string](#) & \_\_s, size\_t \_\_refs=0)
- **\_M\_compare** (const char \*, const char \*) const throw()
- **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()

### 6.303.1 Detailed Description

```
template<typename _CharT>
class std::__cxx11::collate_byname< _CharT >
```

class collate\_byname [22.2.4.2].

### 6.303.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate_byname< _CharT >::char_type
Public typedefs.
```

#### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate_byname< _CharT >::string_type
Public typedefs.
```

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 6.304 std::common\_iterator< \_It, \_Sent > Class Template Reference

```
#include <stl_iterator.h>
```

### Public Member Functions

- constexpr **common\_iterator** (\_It \_\_i) noexcept(is\_nothrow\_move\_constructible\_v< \_It >)
- constexpr **common\_iterator** (\_Sent \_\_s) noexcept(is\_nothrow\_move\_constructible\_v< \_Sent >)
- **common\_iterator** (common\_iterator &&)=default
- constexpr **common\_iterator** (common\_iterator &&\_\_x) noexcept(\_S\_noexcept< \_It, \_Sent >())
- **common\_iterator** (const common\_iterator &)=default
- constexpr **common\_iterator** (const common\_iterator &\_\_x) noexcept(\_S\_noexcept< const \_It &, const \_Sent & >())
- template<typename \_It2, typename \_Sent2>
 constexpr **common\_iterator** (const common\_iterator< \_It2, \_Sent2 > &\_\_x) noexcept(\_S\_noexcept< const \_It2 &, const \_Sent2 & >())
- constexpr decltype(auto) **operator\*** ()
- constexpr decltype(auto) **operator\*** () const
- constexpr common\_iterator & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr auto **operator->** () const
- constexpr common\_iterator & **operator=** (common\_iterator &&)=default
- constexpr common\_iterator & **operator=** (common\_iterator &&\_\_x) noexcept(is\_nothrow\_move\_assignable\_v< \_It > &&is\_nothrow\_move\_assignable\_v< \_Sent > &&is\_nothrow\_move\_constructible\_v< \_It > &&is\_nothrow\_move\_constructible\_v< \_Sent >)
- constexpr common\_iterator & **operator=** (const common\_iterator &)=default
- constexpr common\_iterator & **operator=** (const common\_iterator &\_\_x) noexcept(is\_nothrow\_copy\_assignable\_v< \_It > &&is\_nothrow\_copy\_assignable\_v< \_Sent > &&is\_nothrow\_copy\_constructible\_v< \_It > &&is\_nothrow\_copy\_constructible\_v< \_Sent >)
- template<typename \_It2, typename \_Sent2>
 constexpr common\_iterator & **operator=** (const common\_iterator< \_It2, \_Sent2 > &\_\_x) noexcept(is\_nothrow\_copy\_assignable\_v< \_It, const \_It2 & > &&is\_nothrow\_constructible\_v< \_Sent, const \_Sent2 & > &&is\_nothrow\_assignable\_v< \_It &, const \_It2 & > &&is\_nothrow\_assignable\_v< \_Sent &, const \_Sent2 & >)

**Friends**

- constexpr [iter\\_rvalue\\_reference\\_t](#)< \_It > **iter\_move** (const common\_iterator &\_\_i) noexcept(noexcept(ranges::iter\_move(std::declval< const \_It > >()))))
- template<indirectly\_swappable< \_It > \_It2, typename \_Sent2>  
constexpr void **iter\_swap** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y) noexcept(noexcept(ranges::iter\_swap(std::declval< const \_It > >(), std::declval< const \_It2 > >()))))
- template<sized\_sentinel\_for< \_It > \_It2, sized\_sentinel\_for< \_It > \_Sent2>  
constexpr iter\_difference\_t< \_It2 > **operator-** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)
- template<typename \_It2, sentinel\_for< \_It > \_Sent2>  
constexpr bool **operator==** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)
- template<typename \_It2, sentinel\_for< \_It > \_Sent2>  
constexpr bool **operator==** (const common\_iterator &\_\_x, const common\_iterator< \_It2, \_Sent2 > &\_\_y)

**6.304.1 Detailed Description**

```
template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
requires (!same_as< _It, _Sent>) && copyable< _It>
class std::common_iterator< _It, _Sent >
```

An iterator/sentinel adaptor for representing a non-common range.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**6.305 std::common\_type< \_Tp > Struct Template Reference****6.305.1 Detailed Description**

```
template<typename... _Tp>
struct std::common_type< _Tp >
```

common\_type

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**6.306 std::common\_type< chrono::duration< \_Rep, \_Period > > Struct Template Reference**

```
#include <chrono.h>
```

**Public Types**

- using **type**

**6.306.1 Detailed Description**

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period > >
```

Specialization of common\_type for one chrono::duration type.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 6.307 `std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >` Struct Template Reference

```
#include <chrono.h>
```

#### Public Types

- using `type`

#### 6.307.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
```

Specialization of `common_type` for two identical `chrono::duration` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 6.308 `std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >` Struct Template Reference

```
#include <chrono.h>
```

#### 6.308.1 Detailed Description

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
```

Specialization of `common_type` for `chrono::duration` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 6.309 `std::common_type< chrono::time_point< _Clock, _Duration > >` Struct Template Reference

```
#include <chrono.h>
```

#### Public Types

- using `type`

#### 6.309.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration > >
```

Specialization of `common_type` for one `chrono::time_point` type.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 6.310 `std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >` Struct Template Reference

```
#include <chrono.h>
```

## Public Types

- using [type](#)

### 6.310.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration
> >
```

Specialization of `common_type` for two identical `chrono::time_point` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

## **6.311 std::common\_type< chrono::time\_point< \_Clock, \_Duration1 >, chrono::time\_point< \_Clock, \_Duration2 > > Struct Template Reference**

```
#include <chrono.h>
```

### 6.311.1 Detailed Description

```
template<typename _Clock, typename _Duration1, typename _Duration2>
struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
```

Specialization of `common_type` for `chrono::time_point` types.  
The documentation for this struct was generated from the following file:

- [chrono.h](#)

## **6.312 std::compare\_three\_way\_result< \_Tp, \_Up > Struct Template Reference**

```
#include <compare>
```

### 6.312.1 Detailed Description

```
template<typename _Tp, typename _Up = _Tp>
struct std::compare_three_way_result< _Tp, _Up >
```

[`cmp.result`], result of three-way comparison  
The documentation for this struct was generated from the following file:

- [compare](#)

## **6.313 std::complex< \_Tp > Class Template Reference**

```
#include <complex>
```

## Public Types

- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr `complex` (const `_Tp` &\_\_r=\_Tp(), const `_Tp` &\_\_i=\_Tp())
- constexpr `complex` (const `complex` &)=default
- constexpr `complex` \_\_rep () const
- template<typename `_Up`>  
explicit (!requires(`_Up` \_\_u) { `_Tp`{\_\_u};}) const expr `complex`(const `complex`< `_Up` > &\_\_z)
- constexpr `_Tp` `imag` () const
- constexpr void `imag` (`_Tp` \_\_val)
- constexpr `complex`< `_Tp` > & `operator*=` (const `_Tp` &)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator*=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator+=` (const `_Tp` &\_\_t)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator+=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator-=` (const `_Tp` &\_\_t)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator-=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator/=` (const `_Tp` &)
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator/=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator=` (const `_Tp` &)
- constexpr `complex` & `operator=` (const `complex` &)=default
- template<typename `_Up`>  
constexpr `complex`< `_Tp` > & `operator=` (const `complex`< `_Up` > &)
- constexpr `_Tp` `real` () const
- constexpr void `real` (`_Tp` \_\_val)

### 6.313.1 Detailed Description

```
template<typename _Tp>
class std::complex< _Tp >
```

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

#### Parameters

|                        |                                    |
|------------------------|------------------------------------|
| <code><i>Tp</i></code> | Type of real and imaginary values. |
|------------------------|------------------------------------|

### 6.313.2 Member Typedef Documentation

#### value\_type

```
template<typename _Tp>
typedef _Tp std::complex< _Tp >::value_type
Value typedef.
```

### 6.313.3 Constructor & Destructor Documentation

#### complex()

```
template<typename _Tp>
std::complex< _Tp >::complex (
```



```
const _Tp & __r = _Tp(),
const _Tp & __i = _Tp()) [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

Referenced by [operator\\*=\(\(\)\)](#), [operator\\*=\(\(\)\)](#), [operator+=\(\(\)\)](#), [operator-=\(\(\)\)](#), [operator/=\(\(\)\)](#), [operator/=\(\(\)\)](#), [operator/=\(\(\)\)](#), and [operator/=\(\(\)\)](#).

#### 6.313.4 Member Function Documentation

##### explicit()

```
template<typename _Tp>
template<typename _Up>
std::complex<_Tp>::explicit (
 !requires(_Up __u) { _Tp{__u}; }) const & [inline]
```

Converting constructor.

##### operator\*=(()) [1/2]

```
template<typename _Tp>
complex<_Tp> & std::complex<_Tp>::operator*= (
 const _Tp &) [constexpr]
```

Multiply this complex number by a scalar.

References [complex\(\)](#).

##### operator\*=(()) [2/2]

```
template<typename _Tp>
template<typename _Up>
complex<_Tp> & std::complex<_Tp>::operator*= (
 const complex<_Up> &) [constexpr]
```

Multiply this complex number by another.

References [complex\(\)](#).

##### operator+=(()) [1/2]

```
template<typename _Tp>
complex<_Tp> & std::complex<_Tp>::operator+= (
 const _Tp & __t) [inline], [constexpr]
```

Add a scalar to this complex number.

##### operator+=(()) [2/2]

```
template<typename _Tp>
template<typename _Up>
complex<_Tp> & std::complex<_Tp>::operator+= (
 const complex<_Up> &) [constexpr]
```

Add another complex number to this one.

References [complex\(\)](#).

##### operator-=(()) [1/2]

```
template<typename _Tp>
complex<_Tp> & std::complex<_Tp>::operator-= (
 const _Tp & __t) [inline], [constexpr]
```

Subtract a scalar from this complex number.

**operator-=( ) [2/2]**

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator-= (
 const complex< _Up > &) [constexpr]
```

Subtract another complex number from this one.

References [complex\(\)](#).

**operator/=( ) [1/2]**

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator/= (
 const _Tp &) [constexpr]
```

Divide this complex number by a scalar.

References [complex\(\)](#).

**operator/=( ) [2/2]**

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator/= (
 const complex< _Up > &) [constexpr]
```

Divide this complex number by another.

References [complex\(\)](#).

**operator=( ) [1/2]**

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator= (
 const _Tp &) [constexpr]
```

Assign a scalar to this complex number.

References [complex\(\)](#).

**operator=( ) [2/2]**

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator= (
 const complex< _Up > &) [constexpr]
```

Assign another complex number to this one.

References [complex\(\)](#).

The documentation for this class was generated from the following file:

- [complex](#)

**6.314 std::complex< double > Class Reference**

```
#include <complex>
```

**Public Types**

- typedef \_\_complex\_\_ double **\_ComplexT**
- typedef double **value\_type**

**Public Member Functions**

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (double \_\_r=0.0, double \_\_i=0.0)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr double real() const
- constexpr \_ComplexT **\_\_rep** () const
- template<typename \_Up>  
**explicit** (!requires(\_Up \_\_u) { value\_type{\_\_u};}) const expr complex(const complex< \_Up > &\_\_z)
- constexpr void **imag** (double \_\_val)
- template<typename \_Tp>  
constexpr complex & **operator\*=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator\*=** (double \_\_d)
- template<typename \_Tp>  
constexpr complex & **operator+=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator+=** (double \_\_d)
- template<typename \_Tp>  
constexpr complex & **operator-=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator-=** (double \_\_d)
- template<typename \_Tp>  
constexpr complex & **operator/=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator/=** (double \_\_d)
- constexpr complex & **operator=** (const complex &)=default
- template<typename \_Tp>  
constexpr complex & **operator=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator=** (double \_\_d)
- constexpr void **real** (double \_\_val)

**6.314.1 Detailed Description**

26.2.3 complex specializations complex<double> specialization

The documentation for this class was generated from the following file:

- [complex](#)

**6.315 std::complex< float > Class Reference**

```
#include <complex>
```

**Public Types**

- typedef \_\_complex\_\_ float **\_ComplexT**
- typedef float **value\_type**

**Public Member Functions**

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (float \_\_r=0.0f, float \_\_i=0.0f)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float imag() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float real() const
- constexpr \_ComplexT **\_\_rep** () const

- `template<typename _Up>`  
`explicit (!requires(_Up __u) { value_type{__u};}) const expr complex(const complex< _Up > &__z)`
- `constexpr void imag (float __val)`
- `template<class _Tp>`  
`constexpr complex & operator*= (const complex< _Tp > &__z)`
- `constexpr complex & operator*= (float __f)`
- `template<typename _Tp>`  
`constexpr complex & operator+= (const complex< _Tp > &__z)`
- `constexpr complex & operator+= (float __f)`
- `template<class _Tp>`  
`constexpr complex & operator-= (const complex< _Tp > &__z)`
- `constexpr complex & operator-= (float __f)`
- `template<class _Tp>`  
`constexpr complex & operator/= (const complex< _Tp > &__z)`
- `constexpr complex & operator/= (float __f)`
- `constexpr complex & operator= (const complex &)=default`
- `template<typename _Tp>`  
`constexpr complex & operator= (const complex< _Tp > &__z)`
- `constexpr complex & operator= (float __f)`
- `constexpr void real (float __val)`

### 6.315.1 Detailed Description

26.2.3 complex specializations `complex<float>` specialization

The documentation for this class was generated from the following file:

- [complex](#)

## 6.316 `std::complex< long double >` Class Reference

```
#include <complex>
```

### Public Types

- `typedef __complex__ long double _ComplexT`
- `typedef long double value_type`

### Public Member Functions

- `constexpr complex ( _ComplexT __z)`
- `constexpr complex (const complex &)=default`
- `constexpr complex (long double __r=0.0L, long double __i=0.0L)`
- `__attribute__((abi_tag("cxx11"))) const expr long double imag() const`
- `__attribute__((abi_tag("cxx11"))) const expr long double real() const`
- `constexpr _ComplexT __rep () const`
- `template<typename _Up>`  
`explicit (!requires(_Up __u) { value_type{__u};}) const expr complex(const complex< _Up > &__z)`
- `constexpr void imag (long double __val)`
- `template<typename _Tp>`  
`constexpr complex & operator*= (const complex< _Tp > &__z)`
- `constexpr complex & operator*= (long double __r)`
- `template<typename _Tp>`  
`constexpr complex & operator+= (const complex< _Tp > &__z)`

- constexpr complex & **operator+=** (long double \_\_r)
- template<typename \_Tp>  
constexpr complex & **operator-=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator-=** (long double \_\_r)
- template<typename \_Tp>  
constexpr complex & **operator/=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator/=** (long double \_\_r)
- constexpr complex & **operator=** (const complex &)=default
- template<typename \_Tp>  
constexpr complex & **operator=** (const complex< \_Tp > &\_\_z)
- constexpr complex & **operator=** (long double \_\_r)
- constexpr void **real** (long double \_\_val)

### 6.316.1 Detailed Description

26.2.3 complex specializations complex<long double> specialization

The documentation for this class was generated from the following file:

- [complex](#)

## 6.317 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

```
#include <cond_key_dtor_entry_dealtor.hpp>
```

### Public Types

- typedef HT\_Map::entry **entry**
- typedef HT\_Map::entry\_allocator **entry\_allocator**
- typedef alloc\_traits::allocator\_type **entry\_allocator**
- typedef alloc\_traits::pointer **entry\_pointer**
- typedef HT\_Map::key\_type **key\_type**

### Public Member Functions

- **cond\_dealtor** (entry\_allocator \*p\_a, entry \*p\_e)
- **cond\_dealtor** (entry\_pointer p\_e)
- void **set\_key\_destruct** ()
- void **set\_no\_action** ()
- void **set\_no\_action\_destructor** ()

### Protected Attributes

- bool **m\_key\_destruct**
- entry\_allocator \*const **m\_p\_a**
- entry \*const **m\_p\_e**

### 6.317.1 Detailed Description

```
template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional dey destructor, cc\_hash.

The documentation for this class was generated from the following files:

- [cond\\_dealtor.hpp](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#)

### 6.318 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

```
#include <ov_tree_map_.hpp>
```

#### Public Member Functions

- **cond\_dtor** (value\_vector a\_vec, iterator &r\_last\_it, Size\_Type total\_size)
- void **set\_no\_action** ()

#### Protected Attributes

- value\_vector **m\_a\_vec**
- const Size\_Type **m\_max\_size**
- bool **m\_no\_action**
- iterator & **m\_r\_last\_it**

#### 6.318.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵
Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor<
Size_Type >
```

Conditional destructor.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

### 6.319 `__gnu_cxx::condition_base` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::condition_base`:



#### Public Member Functions

- **condition\_base** (const condition\_base &)=default
- condition\_base & **operator=** (const condition\_base &)=default

### 6.319.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature void throw\_conditionally()

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.320 std::condition\_variable Class Reference

```
#include <condition_variable>
```

### Public Types

- typedef \_\_gthread\_cond\_t \* **native\_handle\_type**

### Public Member Functions

- **condition\_variable** (const condition\_variable &)=delete
- native\_handle\_type **native\_handle** ()
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- condition\_variable & **operator=** (const condition\_variable &)=delete
- void **wait** (unique\_lock< mutex > &\_\_lock)
- template<typename \_Predicate>  
void **wait** (unique\_lock< mutex > &\_\_lock, \_Predicate \_\_p)
- template<typename \_Rep, typename \_Period>  
**cv\_status** **wait\_for** (unique\_lock< mutex > &\_\_lock, const chrono::duration< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Rep, typename \_Period, typename \_Predicate>  
bool **wait\_for** (unique\_lock< mutex > &\_\_lock, const chrono::duration< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Clock, typename \_Duration>  
**cv\_status** **wait\_until** (unique\_lock< mutex > &\_\_lock, const chrono::time\_point< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration, typename \_Predicate>  
bool **wait\_until** (unique\_lock< mutex > &\_\_lock, const chrono::time\_point< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)
- template<typename \_Duration>  
**cv\_status** **wait\_until** (unique\_lock< mutex > &\_\_lock, const chrono::time\_point< steady\_clock, \_Duration > &\_\_atime)
- template<typename \_Duration>  
**cv\_status** **wait\_until** (unique\_lock< mutex > &\_\_lock, const chrono::time\_point< system\_clock, \_Duration > &\_\_atime)

### 6.320.1 Detailed Description

condition\_variable

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 6.321 std::condition\_variable\_any Class Reference

```
#include <condition_variable>
```

## Public Member Functions

- **condition\_variable\_any** (const condition\_variable\_any &)=delete
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- condition\_variable\_any & **operator=** (const condition\_variable\_any &)=delete
- template<typename \_Lock>  
void **wait** (\_Lock &\_\_lock)
- template<typename \_Lock, typename \_Predicate>  
void **wait** (\_Lock &\_\_lock, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Rep, typename \_Period>  
[cv\\_status](#) **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Lock, typename \_Rep, typename \_Period, typename \_Predicate>  
bool **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Lock, typename \_Clock, typename \_Duration>  
[cv\\_status](#) **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Lock, typename \_Clock, typename \_Duration, typename \_Predicate>  
bool **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

### 6.321.1 Detailed Description

condition\_variable\_any

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

## 6.322 std::conditional< \_Cond, \_Iftrue, \_Iffalse > Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using **type**

### 6.322.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional< _Cond, _Iftrue, _Iffalse >
```

Define a member typedef `type` to one of two argument types.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.323 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >::const\_iterator Struct Reference

```
#include <pat_trie_base.hpp>
```



Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator`:



### Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

### Public Member Functions

- **const\_iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (`const const_iterator &other`) const
- `node_const_pointer` **operator\*** () const
- `const_iterator &` **operator++** ()
- `const_iterator` **operator++** (int)
- `const node_pointer_pointer` **operator->** () const
- bool **operator==** (`const const_iterator &other`) const

### Public Attributes

- `node_pointer_pointer` **m\_p\_p\_cur**
- `node_pointer_pointer` **m\_p\_p\_end**

#### 6.323.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator
```

Constant child iterator.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 6.324 std::chrono::tzdb\_list::const\_iterator Class Reference

```
#include <chrono>
```

### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- **const\_iterator** (const const\_iterator &)=default
- **const\_iterator** (const\_iterator &&)=default
- reference **operator\*** () const noexcept
- const\_iterator & **operator++** ()
- const\_iterator **operator++** (int)
- pointer **operator->** () const noexcept
- const\_iterator & **operator=** (const const\_iterator &)=default
- const\_iterator & **operator=** (const\_iterator &&)=default
- bool **operator==** (const const\_iterator &) const noexcept=default

### Friends

- class **tzdb\_list**

#### 6.324.1 Detailed Description

An iterator into the `tzdb_list`

As a extension, in libstdc++ each `tzdb` is reference-counted and the `const_iterator` type shares ownership of the object it refers to. This ensures that a `tzdb` erased from the list will not be destroyed while there is an iterator that refers to it.

The documentation for this class was generated from the following file:

- [chrono](#)

## 6.325 std::const\_mem\_fun1\_ref\_t<\_Ret, \_Tp, \_Arg> Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret` (`_Tp`::\* `__pf`) (`_Arg`) `const`)
- `_Ret operator()` (`const _Tp &__r, _Arg __x`) `const`

#### 6.325.1 Detailed Description

template<typename `_Ret`, typename `_Tp`, typename `_Arg`>  
 class std::const\_mem\_fun1\_ref\_t< `_Ret`, `_Tp`, `_Arg` >

One of the [adaptors for member pointers](#).

#### 6.325.2 Member Typedef Documentation

##### first\_argument\_type

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::first\_argument\_type [inherited]  
`first_argument_type` is the type of the first argument

##### result\_type

typedef `_Ret` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::result\_type [inherited]  
`result_type` is the return type

##### second\_argument\_type

typedef `_Arg` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::second\_argument\_type [inherited]  
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 6.326 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef const \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

### Public Member Functions

- **const\_mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)
- **\_Ret operator()** (const \_Tp \*\_\_p, \_Arg \_\_x) const

#### 6.326.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

#### 6.326.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

typedef \_Arg [std::binary\\_function](#)< const \_Tp \*, \_Arg, \_Ret >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**6.327 std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >:

**Public Types**

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

**Public Member Functions**

- **const\_mem\_fun\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)() const)
- **\_Ret operator()** (const \_Tp &\_\_r) const

**6.327.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**6.327.2 Member Typedef Documentation****argument\_type**

typedef \_Tp [std::unary\\_function](#)< \_Tp, \_Ret >::argument\_type [inherited]  
 argument\_type is the type of the argument

**result\_type**

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**6.328 std::const\_mem\_fun\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:

**Public Types**

- typedef `const _Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

**Public Member Functions**

- `const_mem_fun_t(_Ret(_Tp::*__pf)() const)`
- `_Ret operator() (const _Tp *__p) const`

**6.328.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**6.328.2 Member Typedef Documentation****argument\_type**

```
typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**6.329 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>` Struct Template Reference**

```
#include <functional>
```

**Public Types**

- typedef `_Arg1` **first\_argument\_type**
- typedef `_Result` **result\_type**
- typedef `_Arg2` **second\_argument\_type**

**Public Member Functions**

- **constant\_binary\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** (const `_Arg1` &, const `_Arg2` &) const

**Public Attributes**

- `_Result` **M\_val**

**6.329.1 Detailed Description**

```
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>
```

An [SGI extension](#) .

The documentation for this struct was generated from the following file:

- [ext/functional](#)

**6.330 `__gnu_parallel::constant_size_blocks_tag` Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



### 6.330.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.331 `__gnu_cxx::constant_unary_fun<_Result, _Argument>` Struct Template Reference

```
#include <functional>
```

### Public Types

- typedef `_Argument` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- **constant\_unary\_fun** (`const _Result &__v`)
- `const result_type &` **operator()** (`const _Argument &`) `const`

### Public Attributes

- `result_type` **\_M\_val**

### 6.331.1 Detailed Description

```
template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun<_Result, _Argument>
```

An [SGI extension](#) .

The documentation for this struct was generated from the following file:

- [ext/functional](#)



**6.332 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference**

```
#include <functional>
```

**Public Types**

- typedef `_Result` **result\_type**

**Public Member Functions**

- **constant\_void\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** () const

**Public Attributes**

- `result_type` **\_M\_val**

**6.332.1 Detailed Description**

```
template<class _Result>
struct __gnu_cxx::constant_void_fun<_Result>
```

An [SGI extension](#) .

The documentation for this struct was generated from the following file:

- [ext/functional](#)

**6.333 `__gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_TI>` Struct Template Reference****6.333.1 Detailed Description**

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_TI = null_type>
struct __gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_TI>
```

Dispatch mechanism, primary template for associative types.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

**6.334 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>` Struct Template Reference**

```
#include <priority_queue_base_dispatch.hpp>
```

**Public Types**

- typedef [binary\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc> **type**

**6.334.1 Detailed Description**

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>
```

Specialization for `binary_heap`.

### 6.334.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 6.335 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< \_VTp, Cmp\_Fn, \_Alloc, [binomial\\_heap\\_tag](#), [null\\_type](#) > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

#### Public Types

- typedef [binomial\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 6.335.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
```

Specialization for [binomial\\_heap](#).

### 6.335.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 6.336 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< \_VTp, Cmp\_Fn, \_Alloc, [pairing\\_heap\\_tag](#), [null\\_type](#) > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

#### Public Types

- typedef [pairing\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 6.336.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
```

Specialization for [pairing\\_heap](#).

## 6.336.2 Member Typedef Documentation

### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp,
Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

## 6.337 \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

### Public Types

- typedef [rc\\_binomial\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc > [type](#)

### 6.337.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type
>
```

Specialization for rc\_binary\_heap.

## 6.337.2 Member Typedef Documentation

### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp,
Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

## 6.338 \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

### Public Types

- typedef [thin\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc > [type](#)

### 6.338.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >
```

Specialization for thin\_heap.

### 6.338.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

### 6.339 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [cc\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at2t > [type](#)

#### 6.339.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash map.

### 6.339.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.340 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [gp\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at5t, at2t > [type](#)

#### 6.340.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

### 6.340.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
```

```
typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key,
```

```
Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.341 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [lu\\_map](#)< Key, Mapped, at0t, \_Alloc, at1t > [type](#)

#### 6.341.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
```

```
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update map.

#### 6.341.2 Member Typedef Documentation

##### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
```

```
typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key,
```

```
Mapped, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.342 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [ov\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

#### 6.342.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
```

```
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree map.

### 6.342.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.343 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, [pat\\_trie\\_tag](#), Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [pat\\_trie\\_map](#)< Key, Mapped, at1t, \_Alloc > **type**

#### 6.343.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >
```

Specialization for PATRICIA trie map.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.344 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< Key, Mapped, \_Alloc, [rb\\_tree\\_tag](#), Policy\_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef [rb\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > **type**

#### 6.344.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >
```

Specialization for R-B tree map.

### 6.344.2 Member Typedef Documentation

#### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef rb_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 6.345 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `splay_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

#### 6.345.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree map.

#### 6.345.2 Member Typedef Documentation

##### type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

## 6.346 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` `type`

#### 6.346.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

#### 6.346.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_
Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

### 6.347 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `gp_ht_set< Key, null\_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` [type](#)

#### 6.347.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash set.

#### 6.347.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.348 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `lu_set< Key, null\_type, at0t, _Alloc, at1t >` [type](#)

#### 6.348.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update set.

#### 6.348.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)



## 6.349 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `ov_tree_set< Key, null\_type, at0t, at1t, _Alloc >` [type](#)

#### 6.349.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree set.

#### 6.349.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 6.350 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

### Public Types

- typedef `pat_trie_set< Key, null\_type, at1t, _Alloc >` [type](#)

#### 6.350.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie set.

#### 6.350.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.351 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `rb_tree_set< Key, null\_type, at0t, at1t, _Alloc >` `type`

#### 6.351.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree set.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.352 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

#### Public Types

- typedef `splay_tree_set< Key, null\_type, at0t, at1t, _Alloc >` `type`

#### 6.352.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree set.

#### 6.352.2 Member Typedef Documentation

##### type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 6.353 `__gnu_pbds::container_error` Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for `__gnu_pbds::container_error`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 6.353.1 Detailed Description

Base class for exceptions.

#### 6.353.2 Member Function Documentation

##### **what()**

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 6.354 `__gnu_pbds::container_tag` Struct Reference

```
#include <tag_and_trait.hpp>
```



## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [container\\_traits\\_base](#)< container\_category > **base\_type**
- typedef Cntnr::container\_category **container\_category**
- typedef Cntnr **container\_type**
- typedef base\_type::invalidation\_guarantee **invalidation\_guarantee**

### 6.355.1 Detailed Description

```
template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

### 6.355.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Cntnr>
anonymous enum
```

#### Enumerator

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <a href="#">order_preserving</a>     | True only if Cntnr objects guarantee storing keys by order. |
| <a href="#">erase_can_throw</a>      | True only if erasing a key can throw.                       |
| <a href="#">split_join_can_throw</a> | True only if split or join operations can throw.            |
| <a href="#">reverse_iteration</a>    | True only reverse iterators are supported.                  |

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.356 \_\_gnu\_pbds::container\_traits\_base<\_Tag> Struct Template Reference

### 6.356.1 Detailed Description

```
template<typename _Tag>
struct __gnu_pbds::container_traits_base<_Tag>
```

Primary template, container traits base.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.357 \_\_gnu\_pbds::container\_traits\_base< binary\_heap\_tag > Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [binary\\_heap\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

### 6.357.1 Detailed Description

Specialization, binary heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.358 `__gnu_pbds::container_traits_base< binomial_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `binomial_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

### 6.358.1 Detailed Description

Specialization, binomial heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.359 `__gnu_pbds::container_traits_base< cc_hash_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `cc_hash_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

### 6.359.1 Detailed Description

Specialization, cc hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.360 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `gp_hash_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

### 6.360.1 Detailed Description

Specialization, gp hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.361 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `list_update_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 6.361.1 Detailed Description

Specialization, list update.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.362 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `ov_tree_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

#### 6.362.1 Detailed Description

Specialization, ov tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.363 `__gnu_pbds::container_traits_base< pairing_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef `pairing_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 6.363.1 Detailed Description

Specialization, pairing heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.364 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [pat\\_trie\\_tag](#) [container\\_category](#)
- typedef [range\\_invalidation\\_guarantee](#) [invalidation\\_guarantee](#)

### 6.364.1 Detailed Description

Specialization, pat trie.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.365 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [rb\\_tree\\_tag](#) [container\\_category](#)
- typedef [range\\_invalidation\\_guarantee](#) [invalidation\\_guarantee](#)

### 6.365.1 Detailed Description

Specialization, rb tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.366 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [rc\\_binomial\\_heap\\_tag](#) [container\\_category](#)
- typedef [point\\_invalidation\\_guarantee](#) [invalidation\\_guarantee](#)

### 6.366.1 Detailed Description

Specialization, rc binomial heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.367 `__gnu_pbds::container_traits_base< splay_tree_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

## Public Types

- enum { [order\\_preserving](#) , [erase\\_can\\_throw](#) , [split\\_join\\_can\\_throw](#) , [reverse\\_iteration](#) }
- typedef [splay\\_tree\\_tag](#) [container\\_category](#)
- typedef [range\\_invalidation\\_guarantee](#) [invalidation\\_guarantee](#)



### 6.367.1 Detailed Description

Specialization, splay tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.368 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Reference

```
#include <tag_and_trait.hpp>
```

### Public Types

- enum { `order_preserving` , `erase_can_throw` , `split_join_can_throw` , `reverse_iteration` }
- typedef [thin\\_heap\\_tag](#) `container_category`
- typedef [point\\_invalidation\\_guarantee](#) `invalidation_guarantee`

### 6.368.1 Detailed Description

Specialization, thin heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.369 `std::contiguous_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::contiguous_iterator_tag`:



### 6.369.1 Detailed Description

Contiguous iterators point to objects stored contiguously in memory.  
 The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 6.370 `std::copyable_function<_Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>` Class Template Reference

```
#include <functional>
```

#### Public Types

- using **result\_type**

#### Public Member Functions

- [copyable\\_function](#) () noexcept
- [template<typename \\_Fn, typename \\_Vt = decay\\_t<\\_Fn>>>](#)  
[copyable\\_function](#) (\_Fn &&\_\_f) noexcept(\_S\_nothrow\_init< \_Vt, \_Fn >())

- [copyable\\_function](#) ([copyable\\_function](#) &&\_\_x) noexcept
- [copyable\\_function](#) ([copyable\\_function](#) const &\_\_x)
- `template<typename _Tp, typename... _Args>`  
[copyable\\_function](#) (`in_place_type_t< _Tp >`, `_Args` &&... \_\_args) noexcept(`_S_nothrow_init< _Tp, _Args... >()`)
- `template<typename _Tp, typename _Up, typename... _Args>`  
[copyable\\_function](#) (`in_place_type_t< _Tp >`, [initializer\\_list](#)< `_Up` > \_\_il, `_Args` &&... \_\_args) noexcept(`_S_nothrow_init< _Tp, initializer_list< _Up > &, _Args... >()`)
- [copyable\\_function](#) (`nullptr_t`) noexcept
- `operator bool` () const noexcept
- `_Res operator()` (`_ArgTypes...` \_\_args) `_GLIBCXX_MOF_CV` noexcept(`_Noex`)
- `template<typename _Fn>`  
[copyable\\_function](#) & `operator=` (`_Fn` &&\_\_f) noexcept(`is_nothrow_constructible_v< copyable_function, _Fn >`)
- [copyable\\_function](#) & `operator=` (const [copyable\\_function](#) &\_\_x)
- [copyable\\_function](#) & `operator=` ([copyable\\_function](#) &&\_\_x) noexcept
- [copyable\\_function](#) & `operator=` (`nullptr_t`) noexcept
- `void swap` ([copyable\\_function](#) &\_\_x) noexcept

## Friends

- `template<typename _Func>`  
`constexpr auto & __polyfunc::__base_of` (`_Func` &) noexcept
- `template<typename _Func>`  
`constexpr auto & __polyfunc::__invoker_of` (`_Func` &) noexcept
- `template<typename _Src, typename _Dst>`  
`constexpr bool __polyfunc::__is_invoker_convertible` () noexcept
- `bool operator==` (const [copyable\\_function](#) &\_\_x, `nullptr_t`) noexcept
- `void swap` ([copyable\\_function](#) &\_\_x, [copyable\\_function](#) &\_\_y) noexcept

### 6.370.1 Detailed Description

`template<typename _Res, typename... _ArgTypes, bool _Noex>`  
**class** `std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

Polymorphic copyable function wrapper.

Since

C++26

The `std::copyable_function` class template is a call wrapper similar to `std::function`, but it does not provide information about its target, and preserves constness.

It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `copyable_function::operator()` member function are respected when invoking the target function.

### 6.370.2 Constructor & Destructor Documentation

#### `copyable_function()` [1/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function
() [inline], [noexcept]
```

Creates an empty object.

References [copyable\\_function\(\)](#).

Referenced by [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [copyable\\_function\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator==](#), [swap\(\)](#), and [swap](#).

**copyable\_function()** [2/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 nullptr_t) [inline], [noexcept]
```

Creates an empty object.

References [copyable\\_function\(\)](#).

**copyable\_function()** [3/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.

References [copyable\\_function\(\)](#).

**copyable\_function()** [4/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> const & __x)
[inline]
```

Copies the target object.

References [copyable\\_function\(\)](#).

**copyable\_function()** [5/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

**copyable\_function()** [6/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 in_place_type_t< _Tp > ,
 _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

**copyable\_function()** [7/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
 in_place_type_t< _Tp > ,
 initializer_list< _Up > __il,
 _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [copyable\\_function\(\)](#), and [std::forward\(\)](#).

### 6.370.3 Member Function Documentation

#### `operator bool()`

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

#### `operator()()`

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
 _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `copyable_function` specialization.

#### Precondition

Must not be empty.

References [std::forward\(\)](#), and [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

#### `operator=()` [1/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
 _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References [copyable\\_function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

#### `operator=()` [2/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
 const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline]
```

Stores a copy of the source target object.

References [copyable\\_function\(\)](#), and [operator=\(\)](#).

#### `operator=()` [3/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator= (
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.

References [copyable\\_function\(\)](#), [std::addressof\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**operator=()** [4/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Res) >::operator= (
 nullptr_t) [inline], [noexcept]
```

Destroys the target object (if any).

References [copyable\\_function\(\)](#), and [operator=\(\)](#).

**swap()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Noex) >::swap (
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Noex) > & __x) [inline],
[noexcept]
```

Exchange the target objects (if any).

References [copyable\\_function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#), and [swap](#).

**6.370.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
 const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Noex) > & __x,
 nullptr_t) [friend]
```

Check for emptiness by comparing with `nullptr`.

References [copyable\\_function\(\)](#), and [operator==](#).

Referenced by [operator==](#).

**swap**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Noex) > & __x,
 copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept (_Noex) > & __y) [friend]
```

Exchange the target objects (if any).

References [copyable\\_function\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following file:

- [cpyfunc\\_impl.h](#)

**6.371 std::counted\_iterator< \_It > Class Template Reference**

```
#include <stl_iterator.h>
```

**Public Types**

- using **difference\_type**
- using **iterator\_type**

**Public Member Functions**

- constexpr **counted\_iterator** (**\_It** \_\_i, **iter\_difference\_t**< **\_It** > \_\_n)

- template<typename \_It2>  
constexpr **counted\_iterator** (const counted\_iterator<\_It2> &\_\_x)
- constexpr \_It **base** () &&noexcept(is\_nothrow\_move\_constructible\_v<\_It>)
- constexpr const \_It & **base** () const &noexcept
- constexpr iter\_difference\_t<\_It> **count** () const noexcept
- constexpr decltype(auto) **operator\*** () const noexcept(noexcept(\*\_M\_current))
- constexpr decltype(auto) **operator\*** () noexcept(noexcept(\*\_M\_current))
- constexpr counted\_iterator **operator+** (iter\_difference\_t<\_It> \_\_n) const
- constexpr counted\_iterator & **operator++** ()
- constexpr counted\_iterator **operator++** (int)
- constexpr decltype(auto) **operator++** (int)
- constexpr counted\_iterator & **operator+=** (iter\_difference\_t<\_It> \_\_n)
- constexpr counted\_iterator **operator-** (iter\_difference\_t<\_It> \_\_n) const
- constexpr counted\_iterator & **operator--** ()
- constexpr counted\_iterator **operator--** (int)
- constexpr counted\_iterator & **operator-=** (iter\_difference\_t<\_It> \_\_n)
- constexpr auto **operator->** () const noexcept
- template<typename \_It2>  
constexpr counted\_iterator & **operator=** (const counted\_iterator<\_It2> &\_\_x)
- constexpr decltype(auto) **operator[]** (iter\_difference\_t<\_It> \_\_n) const noexcept(noexcept(\_M\_current[\_\_n]))

## Friends

- constexpr [iter\\_rvalue\\_reference\\_t](#)<\_It> **iter\_move** (const counted\_iterator &\_\_i) noexcept(noexcept(ranges::↵iter\_move(\_\_i.\_M\_current)))
- template<indirectly\_swappable<\_It> \_It2>  
constexpr void **iter\_swap** (const counted\_iterator &\_\_x, const counted\_iterator<\_It2> &\_\_y) noexcept(noexcept(ranges::↵iter\_swap(\_\_x.\_M\_current, \_\_y.\_M\_current)))
- constexpr counted\_iterator **operator+** (iter\_difference\_t<\_It> \_\_n, const counted\_iterator &\_\_x)
- template<common\_with<\_It> \_It2>  
constexpr iter\_difference\_t<\_It2> **operator-** (const counted\_iterator &\_\_x, const counted\_iterator<\_It2> &↵\_\_y) noexcept
- constexpr iter\_difference\_t<\_It> **operator-** (const counted\_iterator &\_\_x, [default\\_sentinel\\_t](#)) noexcept
- constexpr iter\_difference\_t<\_It> **operator-** ([default\\_sentinel\\_t](#), const counted\_iterator &\_\_y) noexcept
- template<common\_with<\_It> \_It2>  
constexpr strong\_ordering **operator<=>** (const counted\_iterator &\_\_x, const counted\_iterator<\_It2> &\_\_y) noexcept
- template<common\_with<\_It> \_It2>  
constexpr bool **operator==** (const counted\_iterator &\_\_x, const counted\_iterator<\_It2> &\_\_y) noexcept
- constexpr bool **operator==** (const counted\_iterator &\_\_x, [default\\_sentinel\\_t](#)) noexcept

### 6.371.1 Detailed Description

template<input\_or\_output\_iterator \_It>  
class std::counted\_iterator<\_It>

An iterator adaptor that keeps track of the distance to the end.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

### 6.372 std::ctype<\_CharT> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype<\_CharT>:



#### Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef \_\_ctype\_abstract\_base<\_CharT>::mask **mask**

#### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool **is** (mask \_\_m, char\_type \_\_c) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const



### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual const char\_type \* [do\\_is](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual char [do\\_narrow](#) (char\_type, char \_\_default) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 6.372.1 Detailed Description

**template<typename \_CharT>**

**class std::ctype<\_CharT>**

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in `__ctype_abstract_base`, to allow for implementation flexibility. See `ctype<wchar_t>` for an example. The functions are documented in `__ctype_abstract_base`.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

## 6.372.2 Member Function Documentation

### do\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

### do\_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

#### Returns

$(M \& \_m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**do\_narrow()** [1/2]

```
template<typename _CharT>
virtual char std::ctype<_CharT>::do_narrow (
 char_type __c,
 char __default) const [protected], [virtual]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, *default* is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted char.

Implements `std::__ctype_abstract_base<_CharT>`.

Referenced by `std::ctype<char>::narrow()`, and `std::ctype<char>::narrow()`.

**do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *\_\_default* is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_scan\_not()**

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_tolower()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_tolower()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::tolower\(\)](#), and [std::ctype<char>::tolower\(\)](#).

**do\_toupper()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

**do\_toupper()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by [std::ctype< char >::toupper\(\)](#), and [std::ctype< char >::toupper\(\)](#).

**do\_widen()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
 char __c) const [protected], [virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

### Parameters

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | The char to convert. |
|-------------------------|----------------------|

### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::widen\(\)](#), and [std::ctype<char>::widen\(\)](#).

### do\_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype<_CharT>::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

### Parameters

|                           |                                   |
|---------------------------|-----------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start range.           |
| $\leftrightarrow$<br>__hi | Pointer to end of range.          |
| $\leftrightarrow$<br>__to | Pointer to the destination array. |

### Returns

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

### is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

#### `is()` [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵type>::do_is()`.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>_↵c</code> | The <code>char_type</code> to compare the mask of. |
| <code>_↵m</code> | The mask to compare against.                       |

#### Returns

`(M & __m) != 0`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

#### `narrow()` [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.



**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted char.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::do\\_get\\_year\(\)](#), [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#), and [std::time\\_put< \\_CharT, \\_OutIter >::put\(\)](#).

**narrow()** [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_not()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**tolower() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**`__hi`.**tolower()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                                       |                           |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>__c</code> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**The lowercase char\_type if convertible, else `__c`.Referenced by `std::time_get<_CharT, _InIter>::get()`.**toupper()** [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**`__hi`.

**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                                                    |                           |
|--------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a> <code>__c</code> | The char_type to convert. |
|--------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Referenced by [std::time\\_get< \\_CharT, \\_Inlter >::get\(\)](#).

**widen()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
 char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                      |                      |
|----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a> <code>__c</code> | The char to convert. |
|----------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted char\_type.

Referenced by [std::money\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::time\\_get< \\_CharT, \\_Inlter >::do\\_get\(\)](#), [std::money\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#) and [std::time\\_put< \\_CharT, \\_Outlter >::do\\_put\(\)](#).

**widen()** [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                          |                                   |
|------------------------------------------------------------------------------------------|-----------------------------------|
| <br>_lo | Pointer to start of range.        |
| <br>_hi | Pointer to end of range.          |
| <br>_to | Pointer to the destination array. |

**Returns**

\_\_hi.

**6.372.3 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id [static]
```

The facet id for ctype<char\_type>.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**6.373 std::ctype< char > Class Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for `std::ctype< char >`:



## Public Types

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef `__ctype_abstract_base< char >::mask` `mask`

## Public Member Functions

- `ctype` (`__c_locale` `__cloc`, const `mask` \* `__table`=0, bool `__del`=false, `size_t` `__refs`=0)
- `ctype` (const `mask` \* `__table`=0, bool `__del`=false, `size_t` `__refs`=0)
- const `char_type` \* `is` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, `mask` \* `__vec`) const
- bool `is` (`mask` `__m`, `char_type` `__c`) const
- const char \* `is` (const char \* `__lo`, const char \* `__hi`, `mask` \* `__vec`) const
- const `char_type` \* `is` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, `mask` \* `__vec`) const
- bool `is` (`mask` `__m`, char `__c`) const
- bool `is` (`mask` `__m`, `char_type` `__c`) const
- char `narrow` (`char_type` `__c`, char `__default`) const
- const `char_type` \* `narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, char `__default`, char \* `__to`) const
- char `narrow` (`char_type` `__c`, char `__default`) const
- char `narrow` (`char_type` `__c`, char `__default`) const
- const `char_type` \* `narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, char `__default`, char \* `__to`) const
- const `char_type` \* `narrow` (const `char_type` \* `__lo`, const `char_type` \* `__hi`, char `__default`, char \* `__to`) const
- const `char_type` \* `scan_is` (`mask` `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- const char \* `scan_is` (`mask` `__m`, const char \* `__lo`, const char \* `__hi`) const

- const [char\\_type](#) \* [scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const char \* [scan\\_not](#) (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const mask \* [table](#) () const throw ()
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const
- const char \* [widen](#) (const char \* \_\_lo, const char \* \_\_hi, [char\\_type](#) \* \_\_to) const

### Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual [~ctype](#) ()
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const =0
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const =0
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual char [do\\_narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const =0

- virtual const `char_type * do_narrow` (const `char_type * __lo`, const `char_type * __hi`, char `__default`, char `* __to`) const =0
- virtual char `do_narrow` (char\_type `__c`, char `__default`) const
- virtual const `char_type * do_narrow` (const `char_type * __lo`, const `char_type * __hi`, char `__default`, char `* __to`) const
- virtual const `char_type * do_scan_is` (mask `__m`, const `char_type * __lo`, const `char_type * __hi`) const =0
- virtual const `char_type * do_scan_is` (mask `__m`, const `char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_scan_not` (mask `__m`, const `char_type * __lo`, const `char_type * __hi`) const =0
- virtual const `char_type * do_scan_not` (mask `__m`, const `char_type * __lo`, const `char_type * __hi`) const
- virtual const `char_type * do_tolower` (char\_type `* __lo`, const `char_type * __hi`) const =0
- virtual `char_type do_tolower` (char\_type `__c`) const =0
- virtual const `char_type * do_tolower` (char\_type `* __lo`, const `char_type * __hi`) const
- virtual `char_type do_tolower` (char\_type `__c`) const
- virtual const `char_type * do_toupper` (char\_type `* __lo`, const `char_type * __hi`) const =0
- virtual `char_type do_toupper` (char\_type `__c`) const =0
- virtual const `char_type * do_toupper` (char\_type `* __lo`, const `char_type * __hi`) const
- virtual `char_type do_toupper` (char\_type `__c`) const
- virtual const char `* do_widen` (const char `* __lo`, const char `* __hi`, `char_type * __to`) const =0
- virtual `char_type do_widen` (char `__c`) const
- virtual const char `* do_widen` (const char `* __lo`, const char `* __hi`, `char_type * __to`) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale & __cloc`) throw ()
- static void `_S_create_c_locale` (`__c_locale & __cloc`, const char `* __s`, `__c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale & __cloc`)
- static `__c_locale _S_get_c_locale` ()
- static const char `* _S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc`, const char `* __s`)

### Protected Attributes

- `__c_locale _M_c_locale_ctype`
- bool `_M_del`
- char `_M_narrow` [1+static\_cast< unsigned char >(-1)]
- char `_M_narrow_ok`
- const mask `* _M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- char `_M_widen` [1+static\_cast< unsigned char >(-1)]
- char `_M_widen_ok`

#### 6.373.1 Detailed Description

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

#### 6.373.2 Member Typedef Documentation

##### char\_type

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter char.



### 6.373.3 Constructor & Destructor Documentation

#### ctype() [1/2]

```
std::ctype< char >::ctype (
 const mask * __table = 0,
 bool __del = false,
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| <code>__table</code> | If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used. |
| <code>__del</code>   | If true, passes ownership of table to this facet.                                           |
| <code>__refs</code>  | Passed to the base facet class.                                                             |

Referenced by [~ctype\(\)](#).

#### ctype() [2/2]

```
std::ctype< char >::ctype (
 __c_locale __cloc,
 const mask * __table = 0,
 bool __del = false,
 size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

#### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__cloc</code>  | Handle to C locale data.                          |
| <code>__table</code> | If non-zero, table is used as the per-char mask.  |
| <code>__del</code>   | If true, passes ownership of table to this facet. |
| <code>__refs</code>  | Passed to the base facet class.                   |

#### ~ctype()

```
virtual std::ctype< char >::~~ctype () [protected], [virtual]
```

Destructor.

This function deletes `table()` if `del` was true in the constructor.

References [ctype\(\)](#).

### 6.373.4 Member Function Documentation

#### classic\_table()

```
const mask * std::ctype< char >::classic_table () throw () [static]
```

Returns a pointer to the C locale mask table.

#### do\_is() [1/3]

```
template<typename _CharT>
```

```
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [pure virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implemented in `std::ctype< _CharT >`, and `std::mask< _CharT >`.

#### do\_is() [2/3]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
 mask __m,
 char_type __c) const [protected], [pure virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

#### Returns

$(M \& \text{__m}) \neq 0$ .

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::is()`, and `std::mask< _CharT >::is()`.

#### do\_is() [3/3]

```
virtual bool std::ctype< char >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| $\leftarrow$<br>_c | The char_type to find the mask of. |
| $\leftarrow$<br>_m | The mask to compare against.       |

#### Returns

(M & \_\_m) != 0.

#### do\_narrow() [1/4]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
 char_type __c,
 char __dfault) const [protected], [pure virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|          |                                     |
|----------|-------------------------------------|
| __c      | The char_type to convert.           |
| __dfault | Char to return if conversion fails. |

#### Returns

The converted char.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::narrow\(\)](#), and [std::mask< \\_CharT >::narrow\(\)](#).

#### do\_narrow() [2/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [protected], [pure virtual], [inherited]
```

Narrow char\_type array to char.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

Implemented in `std::ctype<_CharT>`, and `std::mask<_CharT>`.

#### `do_narrow()` [3/4]

```
virtual char std::ctype< char >::do_narrow (
 char_type __c,
 char __dfault) const [inline], [protected], [virtual]
```

Narrow char.

This virtual function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<char>` facet, `c` will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

#### `do_narrow()` [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline], [protected], [virtual]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

#### do\_scan\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::scan_is()`.

**do\_scan\_is()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [*\_\_lo*,*\_\_hi*) for which is(*\_\_m*,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|             |                              |
|-------------|------------------------------|
| <i>__m</i>  | The mask to compare against. |
| <i>__lo</i> | Pointer to start of range.   |
| <i>__hi</i> | Pointer to end of range.     |

**Returns**

Pointer to a matching char\_type if found, else *\_\_hi*.

**do\_scan\_not()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [*lo*,*hi*) for which is(*m*,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|             |                              |
|-------------|------------------------------|
| <i>__m</i>  | The mask to compare against. |
| <i>__lo</i> | Pointer to start of range.   |
| <i>__hi</i> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else *\_\_hi*.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::scan_not()`.

**do\_scan\_not()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                                                                                           |                              |
|-------------------------------------------------------------------------------------------|------------------------------|
| <br>__m  | The mask to compare against. |
| <br>__lo | Pointer to start of range.   |
| <br>__hi | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else \_\_hi.

**do\_tolower()** [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Implemented in [std::ctype< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_tolower()** [2/4]

```
template<typename _CharT>
```

```
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
 char_type __c) const [protected], [pure virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                          |                           |
|--------------------------|---------------------------|
| $\leftrightarrow$<br>__c | The char_type to convert. |
|--------------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else \_\_c.

Implemented in [std::ctype< \\_CharT >](#), [std::ctype< wchar\\_t >](#), and [std::mask< \\_CharT >](#).

Referenced by [std::mask< \\_CharT >::tolower\(\)](#), and [std::mask< \\_CharT >::do\\_tolower\(\)](#).

#### do\_tolower() [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                           |                                 |
|---------------------------|---------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to first char in range. |
| $\leftrightarrow$<br>__hi | Pointer to end of range.        |

#### Returns

\_\_hi.

References [do\\_tolower\(\)](#).

#### do\_tolower() [4/4]

```
virtual char_type std::ctype< char >::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.



**Parameters**

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__c</code> | The char to convert. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The lowercase char if convertible, else `__c`.

References [do\\_tolower\(\)](#).

Referenced by [do\\_tolower\(\)](#), and [do\\_tolower\(\)](#).

**do\_toupper()** [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implemented in [std::ctype< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_toupper()** [2/4]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
 char_type __c) const [protected], [pure virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                         |                                        |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implemented in `std::ctype< _CharT >`, `std::ctype< wchar_t >`, and `std::mask< _CharT >`.

Referenced by `std::mask< _CharT >::toupper()`, and `std::mask< _CharT >::toupper()`.

**do\_toupper()** [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

References `do_toupper()`.

**do\_toupper()** [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The uppercase char if convertible, else `__c`.

References `do_toupper()`.

Referenced by `do_toupper()`, and `do_toupper()`.

**do\_widen()** [1/3]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [pure virtual], [inherited]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                           |                                   |
|-------------------------------------------------------------------------------------------|-----------------------------------|
| <br>__lo | Pointer to start range.           |
| <br>__hi | Pointer to end of range.          |
| <br>__to | Pointer to the destination array. |

**Returns**

\_\_hi.

Implemented in [std::ctype< \\_CharT >](#), and [std::mask< \\_CharT >](#).

**do\_widen()** [2/3]

```
virtual char_type std::ctype< char >::do_widen (
 char __c) const [inline], [protected], [virtual]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                            |                      |
|--------------------------------------------------------------------------------------------|----------------------|
| <br>__c | The char to convert. |
|--------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted character.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

References [do\\_widen\(\)](#).

Referenced by [do\\_widen\(\)](#).

**do\_widen()** [3/3]

```
virtual const char * std::ctype< char >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [protected], [virtual]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                               |                                   |
|-----------------------------------------------------------------------------------------------|-----------------------------------|
|  <i>__lo</i> | Pointer to start of range.        |
|  <i>__hi</i> | Pointer to end of range.          |
|  <i>__to</i> | Pointer to the destination array. |

**Returns**

*\_\_hi*.

**is()** [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|              |                                      |
|--------------|--------------------------------------|
| <i>__lo</i>  | Pointer to start of range.           |
| <i>__hi</i>  | Pointer to end of range.             |
| <i>__vec</i> | Pointer to an array of mask storage. |

**Returns**

*\_\_hi*.

**is()** [2/6]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

**Parameters**

|                          |                                       |
|--------------------------|---------------------------------------|
| <a href="#">↵</a><br>__c | The char_type to compare the mask of. |
| <a href="#">↵</a><br>__m | The mask to compare against.          |

**Returns**

(M & \_\_m) != 0.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**is()** [3/6]

```
const char * std::ctype< char >::is (
 const char * __lo,
 const char * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

**Returns**

\_\_hi.

**is()** [4/6]

```
const char_type * std::__ctype_abstract_base< char >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

#### is() [5/6]

```
bool std::ctype< char >::is (
 mask __m,
 char __c) const [inline]
```

Test char classification.

This function compares the mask table[c] to `__m`.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

#### Returns

True if `__m & table[__c]` is true, false otherwise.

#### is() [6/6]

```
bool std::__ctype_abstract_base< char >::is (
 mask __m,
 char_type __c) const [inline]
```

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

#### Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__c</code> | The <code>char_type</code> to compare the mask of. |
| <code>__m</code> | The mask to compare against.                       |

**Returns**

(M & \_\_m) != 0.

**narrow()** [1/6]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted char.

Referenced by `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_get< _CharT, _InIter >::get()`, and `std::time_put< _CharT, _OutIter >::put()`.

**narrow()** [2/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *default* is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**narrow()** [3/6]

```
char std::ctype< char >::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(c). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

**Returns**

The converted character.

References [std::ctype<\\_CharT>::do\\_narrow\(\)](#).

**narrow()** [4/6]

```
char std::__ctype_abstract_base< char >::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

**Returns**

The converted char.

**narrow()** [5/6]

```
const char_type * std::ctype< char >::narrow (
 const char_type * __lo,
```



```
const char_type * __hi,
char __dfault,
char * __to) const [inline]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, dfault, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

#### Returns

`__hi`.

References [std::ctype< \\_CharT >::do\\_narrow\(\)](#).

#### narrow() [6/6]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()** [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_is()** [2/3]

```
const char * std::ctype< char >::scan_is (
 mask __m,
 const char * __lo,
 const char * __hi) const [inline]
```

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char if found, else `__hi`.

**scan\_is()** [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                                                                                           |                              |
|-------------------------------------------------------------------------------------------|------------------------------|
| <br>__m  | The mask to compare against. |
| <br>__lo | Pointer to start of range.   |
| <br>__hi | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else \_\_hi.

**scan\_not()** [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                             |                                 |
|---------------------------------------------------------------------------------------------|---------------------------------|
| <br>__m  | The mask to compare against.    |
| <br>__lo | Pointer to first char in range. |
| <br>__hi | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else \_\_hi.

**scan\_not()** [2/3]

```
const char * std::ctype< char >::scan_not (
 mask __m,
```

```
const char * __lo,
const char * __hi) const [inline]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [`__lo`,`__hi`) for which `is(m,char)` is false.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching char if found, else `__hi`.

#### `scan_not()` [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in [`lo`,`hi`) for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

#### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

#### Returns

Pointer to non-matching char if found, else `__hi`.

#### `table()`

```
const mask * std::ctype< char >::table () const throw () [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

#### `tolower()` [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
```

```
char_type * __lo,
const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

#### Parameters

|                                                                                                        |                            |
|--------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"> __lo</a> | Pointer to start of range. |
| <a href="#"> __hi</a> | Pointer to end of range.   |

#### Returns

[!\[\]\(0b5e7e25e8775f7e7e80906ada4f0021\_img.jpg\) \\_\\_hi](#).

#### tolower() [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

#### Parameters

|                                                                                                         |                           |
|---------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"> __c</a> | The char_type to convert. |
|---------------------------------------------------------------------------------------------------------|---------------------------|

#### Returns

The lowercase char\_type if convertible, else \_\_c.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

#### tolower() [3/6]

```
const char_type * std::ctype< char >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_lo, \_\_hi). do\_tolower() must always return the same result for the same input.

#### Parameters

|                                                                                                          |                                 |
|----------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"> __lo</a> | Pointer to first char in range. |
|----------------------------------------------------------------------------------------------------------|---------------------------------|

|                                                                                                                        |                          |
|------------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range. |
|------------------------------------------------------------------------------------------------------------------------|--------------------------|

**Returns**

`__hi`.

References [std::ctype<\\_CharT>::do\\_tolower\(\)](#).

**tolower()** [4/6]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower()** [5/6]

```
char_type std::ctype< char >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the `char` argument to lowercase if possible. If not possible (for example, '2'), returns the argument. `tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                                                                                                         |                      |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__c</code> | The char to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The lowercase char if convertible, else `__c`.

References [std::ctype<\\_CharT>::do\\_tolower\(\)](#).

**tolower()** [6/6]

```
char_type std::__ctype_abstract_base< char >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

|                          |                           |
|--------------------------|---------------------------|
| $\leftrightarrow$<br>__c | The char_type to convert. |
|--------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

**toupper()** [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                           |                            |
|---------------------------|----------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start of range. |
| $\leftrightarrow$<br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

**toupper()** [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                          |                           |
|--------------------------|---------------------------|
| $\leftrightarrow$<br>__c | The char_type to convert. |
|--------------------------|---------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Referenced by [std::time\\_get<\\_CharT, \\_InIter >::get\(\)](#).

**toupper()** [3/6]

```
const char_type * std::ctype< char >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char in the range `[__lo,__hi)` to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(__lo, __hi)`. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

`__hi`.

References [std::ctype<\\_CharT >::do\\_toupper\(\)](#).

**toupper()** [4/6]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [5/6]

```
char_type std::ctype< char >::toupper (
 char_type __c) const [inline]
```



Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument. toupper() acts as if it returns ctype<char>::do\_toupper(c). do\_toupper() must always return the same result for the same input.

#### Parameters

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | The char to convert. |
|-------------------------|----------------------|

#### Returns

The uppercase char if convertible, else \_\_c.

References [std::ctype< \\_CharT >::do\\_toupper\(\)](#).

#### toupper() [6/6]

```
char_type std::__ctype_abstract_base< char >::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

#### Parameters

|                         |                           |
|-------------------------|---------------------------|
| $\leftrightarrow$<br>_c | The char_type to convert. |
|-------------------------|---------------------------|

#### Returns

The uppercase char\_type if convertible, else \_\_c.

#### widen() [1/5]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                          |                            |
|--------------------------|----------------------------|
| $\leftrightarrow$<br>_lo | Pointer to start of range. |
| $\leftrightarrow$<br>_hi | Pointer to end of range.   |

|                                                                                                                                                                                                                                                    |                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <div><div><div><div><div><div><span><span></span></span></div></div></div><div><div><div><span></span></div><div><span><b><code>_↔</code></b></span></div></div></div><div><div><span><b><code>_to</code></b></span></div></div></div></div></div> | Pointer to the destination array. |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|

#### Returns

`__hi`.

#### widen() [2/5]

```
char_type std::ctype< char >::widen (
 char __c) const [inline]
```

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do\_widen(c). do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                                                                                                                                                                   |                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <div><div><div><div><div><div><span><span></span></span></div></div></div><div><div><div><span></span></div><div><span><b><code>_↔</code></b></span></div></div></div><div><div><span><b><code>_c</code></b></span></div></div></div></div></div> | The char to convert. |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|

#### Returns

The converted character.

References [std::ctype< \\_CharT >::do\\_widen\(\)](#).

#### widen() [3/5]

```
char_type std::__ctype_abstract_base< char >::widen (
 char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                                                                                                                                                                   |                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <div><div><div><div><div><div><span><span></span></span></div></div></div><div><div><div><span></span></div><div><span><b><code>_↔</code></b></span></div></div></div><div><div><span><b><code>_c</code></b></span></div></div></div></div></div> | The char to convert. |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|

#### Returns

The converted char\_type.

#### widen() [4/5]

```
const char * std::ctype< char >::widen (
 const char * __lo,
```

```
const char * __hi,
char_type * __to) const [inline]
```

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_widen(c). do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                                     |                                   |
|---------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"> <code>__lo</code></a> | Pointer to first char in range.   |
| <a href="#"> <code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"> <code>__to</code></a> | Pointer to the destination array. |

#### Returns

[!\[\]\(870f5d5e9c0d57485634be3ecf52f3ca\_img.jpg\) `\_\_hi`](#).

References [std::ctype< \\_CharT >::do\\_widen\(\)](#).

#### widen() [5/5]

```
const char * std::__ctype_abstract_base< char >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                                       |                                   |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"> <code>__lo</code></a> | Pointer to start of range.        |
| <a href="#"> <code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"> <code>__to</code></a> | Pointer to the destination array. |

#### Returns

[!\[\]\(aff7c69c44a5e015f18c35867ef3f5c3\_img.jpg\) `\_\_hi`](#).

### 6.373.5 Member Data Documentation

#### id

```
locale::id std::ctype< char >::id [static]
```

The facet id for ctype<char>.

### table\_size

```
const size_t std::ctype< char >::table_size [static]
```

The size of the mask table. It is SCHAR\_MAX + 1.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

## 6.374 std::ctype< wchar\_t > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype< wchar\_t >:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef \_\_ctype\_abstract\_base< wchar\_t >::mask **mask**

### Public Member Functions

- [ctype](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- [ctype](#) (size\_t \_\_refs=0)
- const [char\\_type](#) \* [is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- bool [is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- char [narrow](#) ([char\\_type](#) \_\_c, char \_\_dfault) const
- const [char\\_type](#) \* [narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const [char\\_type](#) \* [scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- const [char\\_type](#) \* [scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const

- const [char\\_type](#) \* [tolower](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- [char\\_type](#) [tolower](#) ([char\\_type](#) \_\_c) const
- const [char\\_type](#) \* [toupper](#) ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- [char\\_type](#) [toupper](#) ([char\\_type](#) \_\_c) const
- [char\\_type](#) [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) **M\_convert\_to\_wmask** (const mask \_\_m) const throw ()
- void **M\_initialize\_ctype** () throw ()
- virtual const [char\\_type](#) \* **do\_is** (const [char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi, mask \*\_\_vec) const
- virtual bool **do\_is** (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual char **do\_narrow** ([char\\_type](#) \_\_c, char \_\_dfault) const
- virtual const [char\\_type](#) \* **do\_narrow** (const [char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual const [char\\_type](#) \* **do\_scan\_is** (mask \_\_m, const [char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual const [char\\_type](#) \* **do\_scan\_not** (mask \_\_m, const [char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual const [char\\_type](#) \* **do\_tolower** ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) **do\_tolower** ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* **do\_toupper** ([char\\_type](#) \*\_\_lo, const [char\\_type](#) \*\_\_hi) const
- virtual [char\\_type](#) **do\_toupper** ([char\\_type](#) \_\_c) const
- virtual [char\\_type](#) **do\_widen** (char \_\_c) const
- virtual const char \* **do\_widen** (const char \*\_\_lo, const char \*\_\_hi, [char\\_type](#) \*\_\_to) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale](#) **S\_clone\_c\_locale** ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void **S\_create\_c\_locale** ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void **S\_destroy\_c\_locale** ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale](#) **S\_get\_c\_locale** ()
- static const char \* **S\_get\_c\_name** () throw ()
- static [\\_\\_c\\_locale](#) **S\_lc\_ctype\_c\_locale** ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

## Protected Attributes

- mask **\_M\_bit** [16]
- **\_\_c\_locale** **\_M\_c\_locale\_ctype**
- char **\_M\_narrow** [128]
- bool **\_M\_narrow\_ok**
- wint\_t **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- **\_\_wmask\_type** **\_M\_wmask** [16]

### 6.374.1 Detailed Description

The ctype<wchar\_t> specialization.

This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well. ctype<wchar\_t> inherits its public methods from \_\_ctype\_abstract\_base<wchar\_t>.

### 6.374.2 Member Typedef Documentation

#### char\_type

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter wchar\_t.

### 6.374.3 Constructor & Destructor Documentation

#### ctype() [1/2]

```
std::ctype< wchar_t >::ctype (
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

#### ctype() [2/2]

```
std::ctype< wchar_t >::ctype (
 __c_locale __cloc,
 size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | Handle to C locale data.        |
| <code>__refs</code> | Passed to the base facet class. |

#### ~ctype()

```
virtual std::ctype< wchar_t >::~~ctype () [protected], [virtual]
```

Destructor.

### 6.374.4 Member Function Documentation

#### do\_is() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each wchar\_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

#### do\_is() [2/2]

```
virtual bool std::ctype< wchar_t >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test wchar\_t classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The wchar_t to find the mask of. |
| <code>__m</code> | The mask to compare against.     |

#### Returns

$(M \ \& \ \_m) \neq 0$ .

Implements `std::__ctype_abstract_base< wchar_t >`.

#### do\_narrow() [1/2]

```
virtual char std::ctype< wchar_t >::do_narrow (
 char_type __c,
 char __default) const [protected], [virtual]
```

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead. For an underived `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__c</code>       | The <code>wchar_t</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.  |

#### Returns

The converted `char`.

Implements [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>](#).

#### `do_narrow()` [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual]
```

Narrow `wchar_t` array to `char` array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, `default` is used instead.

For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to `char`.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

#### Returns

`__hi`.

#### `do_scan_is()`

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
 mask __m,
```



```
const char_type * __lo,
const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t matching mask.

This function searches for and returns the first wchar\_t c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching wchar\_t if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### do\_scan\_not()

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t not matching mask.

This function searches for and returns a pointer to the first wchar\_t c of [\_\_lo,\_\_hi) for which is(\_\_m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a non-matching wchar\_t if found, else \_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### do\_tolower() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

#### Returns

`__hi`.

#### `do_tolower()` [2/2]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

#### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__c</code> | The <code>wchar_t</code> to convert. |
|------------------|--------------------------------------|

#### Returns

The lowercase `wchar_t` if convertible, else `__c`.

Implements `std::__ctype_abstract_base< wchar_t >`.

#### `do_toupper()` [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                          |                            |
|----------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br>_lo | Pointer to start of range. |
| <a href="#"></a><br>_hi | Pointer to end of range.   |

**Returns**

[!\[\]\(a03a7eb2f4046e1d3c76772003e549ea\_img.jpg\)](#)hi.

**do\_toupper()** [2/2]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the wchar\_t argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                         |                         |
|---------------------------------------------------------------------------------------------------------|-------------------------|
| <a href="#"></a><br>_c | The wchar_t to convert. |
|---------------------------------------------------------------------------------------------------------|-------------------------|

**Returns**

The uppercase wchar\_t if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen()** [1/2]

```
virtual char_type std::ctype< wchar_t >::do_widen (
 char __c) const [protected], [virtual]
```

Widen char to wchar\_t.

This virtual function converts the char to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be cast to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                                           |                      |
|-----------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br>_c | The char to convert. |
|-----------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted wchar\_t.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**do\_widen()** [2/2]

```
virtual const char * std::ctype< wchar_t >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [virtual]
```

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                          |                                   |
|--------------------------|-----------------------------------|
| <code><i>__lo</i></code> | Pointer to start range.           |
| <code><i>__hi</i></code> | Pointer to end of range.          |
| <code><i>__to</i></code> | Pointer to the destination array. |

**Returns**

`__hi`.

**is()** [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code><i>__lo</i></code>  | Pointer to start of range.           |
| <code><i>__hi</i></code>  | Pointer to end of range.             |
| <code><i>__vec</i></code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [2/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
```

```
mask __m,
char_type __c) const [inline]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_type>::do\_is().

#### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

#### Returns

(M & \_\_m) != 0.

#### narrow() [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

#### narrow() [2/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_dfault, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.




**scan\_not()**

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                                        |                                 |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**tolower()** [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.


**tolower()** [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                                         |                                        |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

**toupper()** [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

**toupper()** [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

**widen()** [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (
 char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                            |                      |
|--------------------------------------------------------------------------------------------|----------------------|
| <br>__c | The char to convert. |
|--------------------------------------------------------------------------------------------|----------------------|



**Returns**

The converted char\_type.

**widen()** [2/2]

```
const char * std::__ctype_abstract_base< wchar_t >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                           |                                   |
|-------------------------------------------------------------------------------------------|-----------------------------------|
| <br>__lo | Pointer to start of range.        |
| <br>__hi | Pointer to end of range.          |
| <br>__to | Pointer to the destination array. |

**Returns**

\_\_hi.

**6.374.5 Member Data Documentation****id**

```
locale::id std::ctype< wchar_t >::id [static]
```

The facet id for ctype<wchar\_t>.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

**6.375 std::ctype\_base Struct Reference**

```
#include <ctype_base.h>
```

Inheritance diagram for `std::ctype_base`:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### 6.375.1 Detailed Description

Base class for `ctype`.

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

### 6.376 std::ctype\_byname<\_CharT> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype\_byname<\_CharT>:



### Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef ctype<\_CharT>::mask **mask**

### Public Member Functions

- **ctype\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **ctype\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- const char\_type \* **is** (const char\_type \*\_\_lo, const char\_type \*\_\_hi, mask \*\_\_vec) const
- bool **is** (mask \_\_m, char\_type \_\_c) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- const char\_type \* **tolower** (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual const char\_type \* [do\\_is](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual char [do\\_narrow](#) (char\_type, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 6.376.1 Detailed Description

```
template<typename _CharT>
class std::ctype_byname< _CharT >
```

class ctype\_byname [22.2.1.2].

### 6.376.2 Member Function Documentation

#### do\_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

#### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### do\_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype<_CharT>::do_is (
 mask __m,
 char_type __c) const [protected], [virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

#### Returns

$(M \& \_m) \neq 0$ .

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_narrow()** [1/2]

```
template<typename _CharT>
virtual char std::ctype<_CharT>::do_narrow (
 char_type __c,
 char __default) const [protected], [virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, default is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::narrow\(\)](#), and [std::ctype<char>::narrow\(\)](#).

**do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual], [inherited]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).**do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to a matching char\_type if found, else `__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).**do\_scan\_not()**

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which `is(m,c)` is false.`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching `char_type` if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_tolower()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_tolower()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__c</code> | The <code>char_type</code> to convert. |
|------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::tolower\(\)](#), and [std::ctype<char>::tolower\(\)](#).



**do\_toupper()** [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype<_CharT>::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**do\_toupper()** [2/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_toupper (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::toupper\(\)](#), and [std::ctype<char>::toupper\(\)](#).

**do\_widen()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_widen (
 char __c) const [protected], [virtual], [inherited]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | The char to convert. |
|-------------------------|----------------------|

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::widen\(\)](#), and [std::ctype<char>::widen\(\)](#).

#### do\_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype<_CharT>::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                           |                                   |
|---------------------------|-----------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start range.           |
| $\leftrightarrow$<br>__hi | Pointer to end of range.          |
| $\leftrightarrow$<br>__to | Pointer to the destination array. |

#### Returns

\_\_hi.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

### Returns

`__hi`.

### is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_type>::do\_is().

### Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

### Returns

$(M \& \text{__m}) \neq 0$ .

Referenced by [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#).

### narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted char.

Referenced by `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_get<_CharT, _InIter>::get()`, and `std::time_put<_CharT, _OutIter>::put()`.

**narrow()** [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each `char_type` in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_not()**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**tolower() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base<_CharT>::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower() [2/2]**

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                                       |                           |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>__c</code> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**toupper() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**toupper()** [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                   |                           |
|-----------------------------------------------------------------------------------|---------------------------|
|  | The char_type to convert. |
| <code>__c</code>                                                                  |                           |

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Referenced by [std::time\\_get<\\_CharT, \\_Inlter>::get\(\)](#).

**widen()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
 char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                     |                      |
|-------------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                    |                      |

**Returns**

The converted char\_type.

Referenced by [std::money\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::do\\_get\(\)](#), [std::money\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#), and [std::time\\_put<\\_CharT, \\_Outlter>::do\\_put\(\)](#).

**widen()** [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base<_CharT>::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                       |                                   |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"></a><br><code>_lo</code> | Pointer to start of range.        |
| <a href="#"></a><br><code>_hi</code> | Pointer to end of range.          |
| <a href="#"></a><br><code>_to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

**6.376.3 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id [static], [inherited]
The facet id for ctype<char_type>.
The documentation for this class was generated from the following file:
```

- [locale\\_facets.h](#)

**6.377 std::ctype\_byname< char > Class Reference**

```
#include <locale_facets.h>
```



Inheritance diagram for std::ctype\_byname< char >:



### Public Types

- typedef const int \* `__to_type`
- typedef char `char_type`
- typedef `ctype< char >::mask` `mask`

### Public Member Functions

- `ctype_byname` (const char \* \_\_s, size\_t \_\_refs=0)
- `ctype_byname` (const string & \_\_s, size\_t \_\_refs=0)
- const char\_type \* `is` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool `is` (mask \_\_m, char\_type \_\_c) const
- const char\_type \* `is` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- bool `is` (mask \_\_m, char\_type \_\_c) const
- char `narrow` (char\_type \_\_c, char \_\_default) const
- const char\_type \* `narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- char `narrow` (char\_type \_\_c, char \_\_default) const
- const char\_type \* `narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char\_type \* `scan_is` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_is` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_not` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* `scan_not` (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const mask \* `table` () const throw ()
- const char\_type \* `tolower` (char\_type \* \_\_lo, const char\_type \* \_\_hi) const

- char\_type [tolower](#) (char\_type \_\_c) const
- const char\_type \* [tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type [tolower](#) (char\_type \_\_c) const
- const char\_type \* [toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type [toupper](#) (char\_type \_\_c) const
- const char\_type \* [toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- char\_type [toupper](#) (char\_type \_\_c) const
- const char \* [widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const
- char\_type [widen](#) (char \_\_c) const
- const char \* [widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const

### Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual const char\_type \* [do\\_is](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, mask \*\_\_vec) const
- virtual bool [do\\_is](#) (mask \_\_m, char\_type \_\_c) const
- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual char [do\\_narrow](#) (char\_type, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual const char\_type \* [do\\_scan\\_is](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual const char\_type \* [do\\_scan\\_not](#) (mask \_\_m, const char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const

- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_dest) const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale [\\_M\\_c\\_locale\\_ctype](#)
- bool [\\_M\\_del](#)
- char [\\_M\\_narrow](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_narrow\\_ok](#)
- const mask \* [\\_M\\_table](#)
- \_\_to\_type [\\_M\\_tolower](#)
- \_\_to\_type [\\_M\\_toupper](#)
- char [\\_M\\_widen](#) [1+static\_cast< unsigned char >(-1)]
- char [\\_M\\_widen\\_ok](#)

### 6.377.1 Detailed Description

22.2.1.4 Class ctype\_byname specializations.

### 6.377.2 Member Function Documentation

#### classic\_table()

```
const mask * std::ctype< char >::classic_table () throw () [static], [inherited]
```

Returns a pointer to the C locale mask table.

#### do\_is() [1/3]

```
virtual bool std::ctype< char >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

### Parameters

|                     |                                    |
|---------------------|------------------------------------|
| <a href="#">__c</a> | The char_type to find the mask of. |
|---------------------|------------------------------------|

|                          |                              |
|--------------------------|------------------------------|
| <code>__↵<br/>__m</code> | The mask to compare against. |
|--------------------------|------------------------------|

**Returns**

(M & \_\_m) != 0.

**do\_is()** [2/3]

```
virtual const char_type * std::ctype< char >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**do\_is()** [3/3]

```
virtual bool std::ctype< char >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test char\_type classification.

This function finds a mask M for c and compares it to mask m.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__↵<br/>__c</code> | The char_type to find the mask of. |
| <code>__↵<br/>__m</code> | The mask to compare against.       |

**Returns**

(M & \_\_m) != 0.

**do\_narrow()** [1/4]

```
virtual char std::ctype< char >::do_narrow (
 char_type __c,
 char __dfault) const [inline], [protected], [virtual], [inherited]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

**Returns**

The converted char.

**do\_narrow()** [2/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, dfault is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**

`__hi`.

**do\_narrow()** [3/4]

```
virtual char std::ctype< char >::do_narrow (
 char_type ,
 char __default) const [protected], [virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

**do\_narrow()** [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**`__hi`.**do\_scan\_is()** [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to a matching char\_type if found, else `__hi`.**do\_scan\_is()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which is(`__m`,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to a matching char\_type if found, else `__hi`.

**do\_scan\_not()** [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                                                                                           |                              |
|-------------------------------------------------------------------------------------------|------------------------------|
| <br>__m  | The mask to compare against. |
| <br>__lo | Pointer to start of range.   |
| <br>__hi | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else \_\_hi.

**do\_scan\_not()** [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

**Parameters**

|                                                                                             |                              |
|---------------------------------------------------------------------------------------------|------------------------------|
| <br>__m  | The mask to compare against. |
| <br>__lo | Pointer to start of range.   |
| <br>__hi | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else \_\_hi.

**do\_tolower()** [1/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
```



```
char_type * __lo,
const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                                                                                           |                                 |
|-------------------------------------------------------------------------------------------|---------------------------------|
| <br>__lo | Pointer to first char in range. |
| <br>__hi | Pointer to end of range.        |

#### Returns

\_\_hi.

References [do\\_tolower\(\)](#).

#### do\_tolower() [2/4]

```
virtual char_type std::ctype< char >::do_tolower (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

#### Parameters

|                                                                                            |                      |
|--------------------------------------------------------------------------------------------|----------------------|
| <br>__c | The char to convert. |
|--------------------------------------------------------------------------------------------|----------------------|

#### Returns

The lowercase char if convertible, else \_\_c.

References [do\\_tolower\(\)](#).

Referenced by [do\\_tolower\(\)](#), and [do\\_tolower\(\)](#).

#### do\_tolower() [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**do\_tolower() [4/4]**

```
virtual char_type std::ctype< char >::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                                       |                           |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>__c</code> | The char_type to convert. |
|-----------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

**do\_toupper() [1/4]**

```
virtual const char_type * std::ctype< char >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

References [do\\_toupper\(\)](#).

**do\_toupper()** [2/4]

```
virtual char_type std::ctype< char >::do_toupper (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                          |                      |
|--------------------------|----------------------|
| $\leftrightarrow$<br>__c | The char to convert. |
|--------------------------|----------------------|

**Returns**

The uppercase char if convertible, else \_\_c.

References [do\\_toupper\(\)](#).

Referenced by [do\\_toupper\(\)](#), and [do\\_toupper\(\)](#).

**do\_toupper()** [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                           |                            |
|---------------------------|----------------------------|
| $\leftrightarrow$<br>__lo | Pointer to start of range. |
| $\leftrightarrow$<br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

**do\_toupper()** [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char\_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                                                                                                                                        |                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"></a> <code>_c</code> | The char_type to convert. |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

**do\_widen()** [1/3]

```
virtual const char * std::ctype< char >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                                                                                                                              |                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"></a><br><a href="#"></a> <code>__lo</code>  | Pointer to start of range.        |
| <a href="#"></a><br><a href="#"></a> <code>__hi</code> | Pointer to end of range.          |
| <a href="#"></a><br><a href="#"></a> <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

**do\_widen()** [2/3]

```
virtual char_type std::ctype< char >::do_widen (
 char __c) const [protected], [virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                                                                                                                            |                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><a href="#"></a> <code>_c</code> | The char to convert. |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted char\_type

**do\_widen()** [3/3]

```
virtual const char * std::ctype< char >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __dest) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start range.           |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

**is()** [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

**is()** [2/4]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for \_\_c and compares it to mask \_\_m. It does so by returning the value of ctype<char\_↵type>::do\_is().

**Parameters**

|                          |                                       |
|--------------------------|---------------------------------------|
| <a href="#">↵</a><br>__c | The char_type to compare the mask of. |
| <a href="#">↵</a><br>__m | The mask to compare against.          |

**Returns**

(M & \_\_m) != 0.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

**is()** [3/4]

```
const char_type * std::__ctype_abstract_base< char >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

**Returns**

\_\_hi.

**is()** [4/4]

```
bool std::__ctype_abstract_base< char >::is (
 mask __m,
 char_type __c) const [inline]
```

Test char\_type classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵type>::do_is()`.

#### Parameters

|                   |                                       |
|-------------------|---------------------------------------|
| <code>↵__c</code> | The char_type to compare the mask of. |
| <code>↵__m</code> | The mask to compare against.          |

#### Returns

`(M & __m) != 0.`

#### narrow() [1/4]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

#### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Referenced by `std::time_get< _CharT, _Inlter >::do_get_year()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

#### narrow() [2/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**

`__hi`.

**narrow()** [3/4]

```
char std::__ctype_abstract_base< char >::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

**narrow()** [4/4]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline]
```

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.



**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

**scan\_is() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_is() [2/2]**

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching char\_type if found, else `__hi`.

**scan\_not() [1/2]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**scan\_not() [2/2]**

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__m</code>  | The mask to compare against.    |
| <code>__lo</code> | Pointer to first char in range. |
| <code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

**table()**

```
const mask * std::ctype< char >::table () const throw () [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

**tolower()** [1/4]

```
template<typename _CharT>
```

```
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**tolower()** [2/4]

```
template<typename _CharT>
```

```
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                                                |                           |
|--------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"><u>_c</u></a> | The char_type to convert. |
|--------------------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Referenced by [std::time\\_get<\\_CharT, \\_InIter>::get\(\)](#).

**tolower() [3/4]**

```
const char_type * std::__ctype_abstract_base< char >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                                                                                                                 |                            |
|---------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><a href="#"><u>_lo</u></a> | Pointer to start of range. |
| <a href="#"></a><br><a href="#"><u>_hi</u></a> | Pointer to end of range.   |

**Returns**

[\\_\\_hi](#).

**tolower() [4/4]**

```
char_type std::__ctype_abstract_base< char >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

|                                                                                                                                  |                           |
|----------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"><u>_c</u></a> | The char_type to convert. |
|----------------------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

**toupper() [1/4]**

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
```

```
char_type * __lo,
const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

#### Parameters

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

#### Returns

\_\_hi.

#### toupper() [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

#### Parameters

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

#### Returns

The uppercase char\_type if convertible, else \_\_c.

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\(\)](#).

#### toupper() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

#### Parameters

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
|---------------------------------------------------------------------------------------------|----------------------------|

|                                                                                                                                  |                          |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range. |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------------|

**Returns**

[\\_\\_hi](#).

**toupper()** [4/4]

```
char_type std::__ctype_abstract_base< char >::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                                                                 |                           |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><a href="#"><u>__c</u></a> | The char_type to convert. |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else [\\_\\_c](#).

**widen()** [1/3]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                                                                                                                                    |                                   |
|------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#"></a><br><a href="#"><u>__lo</u></a> | Pointer to start of range.        |
| <a href="#"></a><br><a href="#"><u>__hi</u></a> | Pointer to end of range.          |
| <a href="#"></a><br><a href="#"><u>__to</u></a> | Pointer to the destination array. |

**Returns**

[\\_\\_hi](#).

**widen()** [2/3]

```
char_type std::__ctype_abstract_base< char >::widen (
 char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                         |                      |
|-------------------------|----------------------|
| $\hookleftarrow$<br>__c | The char to convert. |
|-------------------------|----------------------|

**Returns**

The converted char\_type.

**widen()** [3/3]

```
const char * std::__ctype_abstract_base< char >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                          |                                   |
|--------------------------|-----------------------------------|
| $\hookleftarrow$<br>__lo | Pointer to start of range.        |
| $\hookleftarrow$<br>__hi | Pointer to end of range.          |
| $\hookleftarrow$<br>__to | Pointer to the destination array. |

**Returns**

\_\_hi.

**6.377.3 Member Data Documentation****id**

```
locale::id std::ctype< char >::id [static]
```

The facet id for ctype<char\_type>.

**table\_size**

```
const size_t std::ctype< char >::table_size [static], [inherited]
```

The size of the mask table. It is `SCHAR_MAX + 1`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 6.378 `std::ranges::dangling` Struct Reference

```
#include <ranges_base.h>
```

### Public Member Functions

- `template<typename... _Args>`  
`constexpr dangling (_Args &&...) noexcept`

#### 6.378.1 Detailed Description

Type returned by algorithms instead of a dangling iterator or subrange.

The documentation for this struct was generated from the following file:

- [ranges\\_base.h](#)

## 6.379 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

```
#include <debug_allocator.h>
```

### Public Types

- `typedef _Traits::const_pointer const_pointer`
- `typedef _Traits::const_reference const_reference`
- `typedef _Traits::difference_type difference_type`
- `typedef _Traits::pointer pointer`
- `typedef _Traits::reference reference`
- `typedef _Traits::size_type size_type`
- `typedef _Traits::value_type value_type`

### Public Member Functions

- `debug_allocator (const _Alloc &__a)`
- `template<typename _Alloc2>`  
`debug_allocator (const debug_allocator<_Alloc2> &__a2, typename __convertible<_Alloc2>::__type=0)`
- `pointer allocate (size_type __n)`
- `pointer allocate (size_type __n, const void *__hint)`
- `template<typename _Tp, typename... _Args>`  
`void construct (_Tp *__p, _Args &&... __args)`
- `void construct (pointer __p, const value_type &__val)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Tp>`  
`void destroy (_Tp *__p)`
- `size_type max_size () const throw ()`

### Friends

- `template<typename _Alloc2>`  
`bool operator== (const debug_allocator &__lhs, const debug_allocator<_Alloc2> &__rhs) noexcept`



### 6.379.1 Detailed Description

```
template<typename _Alloc>
class __gnu_cxx::debug_allocator< _Alloc >
```

A meta-allocator with debugging bits.

This is precisely the allocator defined in the C++03 Standard.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 6.380 std::decimal::decimal128 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat128**

### Public Member Functions

- [decimal128](#) (\_\_decfloat128 \_\_z)
- **decimal128** ([decimal32](#) d32)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (double \_\_r)
- **decimal128** (float \_\_r)
- **decimal128** (int \_\_z)
- **decimal128** (long \_\_z)
- **decimal128** (long double \_\_r)
- **decimal128** (long long \_\_z)
- **decimal128** (unsigned int \_\_z)
- **decimal128** (unsigned long \_\_z)
- **decimal128** (unsigned long long \_\_z)
- \_\_decfloat128 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat128 \_\_x)
- **operator long long** () const
- decimal128 & **operator\*=** (decimal128 \_\_rhs)
- decimal128 & **operator\*=** ([decimal32](#) \_\_rhs)
- decimal128 & **operator\*=** ([decimal64](#) \_\_rhs)
- decimal128 & **operator\*=** (int \_\_rhs)
- decimal128 & **operator\*=** (long \_\_rhs)
- decimal128 & **operator\*=** (long long \_\_rhs)
- decimal128 & **operator\*=** (unsigned int \_\_rhs)
- decimal128 & **operator\*=** (unsigned long \_\_rhs)
- decimal128 & **operator\*=** (unsigned long long \_\_rhs)
- decimal128 & **operator++** ()
- decimal128 **operator++** (int)
- decimal128 & **operator+=** (decimal128 \_\_rhs)
- decimal128 & **operator+=** ([decimal32](#) \_\_rhs)
- decimal128 & **operator+=** ([decimal64](#) \_\_rhs)
- decimal128 & **operator+=** (int \_\_rhs)
- decimal128 & **operator+=** (long \_\_rhs)
- decimal128 & **operator+=** (long long \_\_rhs)

- decimal128 & **operator+=** (unsigned int \_\_rhs)
- decimal128 & **operator+=** (unsigned long \_\_rhs)
- decimal128 & **operator+=** (unsigned long long \_\_rhs)
- decimal128 & **operator--** ()
- decimal128 **operator--** (int)
- decimal128 & **operator-=** (decimal128 \_\_rhs)
- decimal128 & **operator-=** ([decimal32](#) \_\_rhs)
- decimal128 & **operator-=** ([decimal64](#) \_\_rhs)
- decimal128 & **operator-=** (int \_\_rhs)
- decimal128 & **operator-=** (long \_\_rhs)
- decimal128 & **operator-=** (long long \_\_rhs)
- decimal128 & **operator-=** (unsigned int \_\_rhs)
- decimal128 & **operator-=** (unsigned long \_\_rhs)
- decimal128 & **operator-=** (unsigned long long \_\_rhs)
- decimal128 & **operator/=** (decimal128 \_\_rhs)
- decimal128 & **operator/=** ([decimal32](#) \_\_rhs)
- decimal128 & **operator/=** ([decimal64](#) \_\_rhs)
- decimal128 & **operator/=** (int \_\_rhs)
- decimal128 & **operator/=** (long \_\_rhs)
- decimal128 & **operator/=** (long long \_\_rhs)
- decimal128 & **operator/=** (unsigned int \_\_rhs)
- decimal128 & **operator/=** (unsigned long \_\_rhs)
- decimal128 & **operator/=** (unsigned long long \_\_rhs)

### 6.380.1 Detailed Description

3.2.4 Class decimal128.

### 6.380.2 Constructor & Destructor Documentation

#### decimal128()

```
std::decimal::decimal128::decimal128 (
 __decfloat128 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 6.381 std::decimal::decimal32 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat32**

## Public Member Functions

- [decimal32](#) (\_\_decfloat32 \_\_z)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** (double \_\_r)
- **decimal32** (float \_\_r)
- **decimal32** (int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (long double \_\_r)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (unsigned long long \_\_z)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- **operator long long** () const
- decimal32 & **operator\*=** ([decimal128](#) \_\_rhs)
- decimal32 & **operator\*=** (decimal32 \_\_rhs)
- decimal32 & **operator\*=** ([decimal64](#) \_\_rhs)
- decimal32 & **operator\*=** (int \_\_rhs)
- decimal32 & **operator\*=** (long \_\_rhs)
- decimal32 & **operator\*=** (long long \_\_rhs)
- decimal32 & **operator\*=** (unsigned int \_\_rhs)
- decimal32 & **operator\*=** (unsigned long \_\_rhs)
- decimal32 & **operator\*=** (unsigned long long \_\_rhs)
- decimal32 & **operator++** ()
- decimal32 **operator++** (int)
- decimal32 & **operator+=** ([decimal128](#) \_\_rhs)
- decimal32 & **operator+=** (decimal32 \_\_rhs)
- decimal32 & **operator+=** ([decimal64](#) \_\_rhs)
- decimal32 & **operator+=** (int \_\_rhs)
- decimal32 & **operator+=** (long \_\_rhs)
- decimal32 & **operator+=** (long long \_\_rhs)
- decimal32 & **operator+=** (unsigned int \_\_rhs)
- decimal32 & **operator+=** (unsigned long \_\_rhs)
- decimal32 & **operator+=** (unsigned long long \_\_rhs)
- decimal32 & **operator--** ()
- decimal32 **operator--** (int)
- decimal32 & **operator-=** ([decimal128](#) \_\_rhs)
- decimal32 & **operator-=** (decimal32 \_\_rhs)
- decimal32 & **operator-=** ([decimal64](#) \_\_rhs)
- decimal32 & **operator-=** (int \_\_rhs)
- decimal32 & **operator-=** (long \_\_rhs)
- decimal32 & **operator-=** (long long \_\_rhs)
- decimal32 & **operator-=** (unsigned int \_\_rhs)
- decimal32 & **operator-=** (unsigned long \_\_rhs)
- decimal32 & **operator-=** (unsigned long long \_\_rhs)
- decimal32 & **operator/=** ([decimal128](#) \_\_rhs)
- decimal32 & **operator/=** (decimal32 \_\_rhs)
- decimal32 & **operator/=** ([decimal64](#) \_\_rhs)

- decimal32 & **operator**/= (int \_\_rhs)
- decimal32 & **operator**/= (long \_\_rhs)
- decimal32 & **operator**/= (long long \_\_rhs)
- decimal32 & **operator**/= (unsigned int \_\_rhs)
- decimal32 & **operator**/= (unsigned long \_\_rhs)
- decimal32 & **operator**/= (unsigned long long \_\_rhs)

### 6.381.1 Detailed Description

3.2.2 Class decimal32.

### 6.381.2 Constructor & Destructor Documentation

#### decimal32()

```
std::decimal::decimal32::decimal32 (
 __decfloat32 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 6.382 std::decimal::decimal64 Class Reference

```
#include <decimal>
```

### Public Types

- typedef float **\_\_decfloat64**

### Public Member Functions

- [decimal64](#) (\_\_decfloat64 \_\_z)
- **decimal64** ([decimal128](#) d128)
- **decimal64** ([decimal32](#) d32)
- **decimal64** (double \_\_r)
- **decimal64** (float \_\_r)
- **decimal64** (int \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (long double \_\_r)
- **decimal64** (long long \_\_z)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (unsigned long long \_\_z)
- **\_\_decfloat64** **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- **operator long long** () const
- decimal64 & **operator**\*= ([decimal128](#) \_\_rhs)
- decimal64 & **operator**\*= ([decimal32](#) \_\_rhs)
- decimal64 & **operator**\*= (decimal64 \_\_rhs)
- decimal64 & **operator**\*= (int \_\_rhs)
- decimal64 & **operator**\*= (long \_\_rhs)
- decimal64 & **operator**\*= (long long \_\_rhs)
- decimal64 & **operator**\*= (unsigned int \_\_rhs)

- decimal64 & **operator**\*= (unsigned long \_\_rhs)
- decimal64 & **operator**\*= (unsigned long long \_\_rhs)
- decimal64 & **operator**++ ()
- decimal64 **operator**++ (int)
- decimal64 & **operator**+= (decimal128 \_\_rhs)
- decimal64 & **operator**+= (decimal32 \_\_rhs)
- decimal64 & **operator**+= (decimal64 \_\_rhs)
- decimal64 & **operator**+= (int \_\_rhs)
- decimal64 & **operator**+= (long \_\_rhs)
- decimal64 & **operator**+= (long long \_\_rhs)
- decimal64 & **operator**+= (unsigned int \_\_rhs)
- decimal64 & **operator**+= (unsigned long \_\_rhs)
- decimal64 & **operator**+= (unsigned long long \_\_rhs)
- decimal64 & **operator**-- ()
- decimal64 **operator**-- (int)
- decimal64 & **operator**-= (decimal128 \_\_rhs)
- decimal64 & **operator**-= (decimal32 \_\_rhs)
- decimal64 & **operator**-= (decimal64 \_\_rhs)
- decimal64 & **operator**-= (int \_\_rhs)
- decimal64 & **operator**-= (long \_\_rhs)
- decimal64 & **operator**-= (long long \_\_rhs)
- decimal64 & **operator**-= (unsigned int \_\_rhs)
- decimal64 & **operator**-= (unsigned long \_\_rhs)
- decimal64 & **operator**-= (unsigned long long \_\_rhs)
- decimal64 & **operator**/= (decimal128 \_\_rhs)
- decimal64 & **operator**/= (decimal32 \_\_rhs)
- decimal64 & **operator**/= (decimal64 \_\_rhs)
- decimal64 & **operator**/= (int \_\_rhs)
- decimal64 & **operator**/= (long \_\_rhs)
- decimal64 & **operator**/= (long long \_\_rhs)
- decimal64 & **operator**/= (unsigned int \_\_rhs)
- decimal64 & **operator**/= (unsigned long \_\_rhs)
- decimal64 & **operator**/= (unsigned long long \_\_rhs)

### 6.382.1 Detailed Description

3.2.3 Class decimal64.

### 6.382.2 Constructor & Destructor Documentation

#### decimal64()

```
std::decimal::decimal64::decimal64 (
 __decfloat64 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

The documentation for this class was generated from the following file:

- [decimal](#)

## 6.383 parallelism\_v2::simd\_abi::deduce<\_Tp, \_Np,...> Struct Template Reference

```
#include <simd.h>
```

### 6.383.1 Detailed Description

```
template<typename _Tp, size_t _Np, typename...>
struct parallelism_v2::simd_abi::deduce< _Tp, _Np,... >
```

#### Template Parameters

|                    |                                                                                                                                                                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Tp</code>   | The requested <code>value_type</code> for the elements.                                                                                                                                                                                                                                       |
| <code>_Np</code>   | The requested number of elements.                                                                                                                                                                                                                                                             |
| <code>_Abis</code> | This parameter is ignored, since this implementation cannot make any use of it. Either <code>__a</code> good native ABI is matched and used as <code>type alias</code> , or the <code>fixed_size&lt;_Np&gt;</code> ABI is used, which internally is built from the best matching native ABIs. |

The documentation for this struct was generated from the following file:

- `simd.h`

### 6.384 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef `direct_mask_range_hashing` type

#### 6.384.1 Detailed Description

Primary template, `default_comb_hash_fn`.

#### 6.384.2 Member Typedef Documentation

##### type

```
typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- `standard_policies.hpp`

### 6.385 `std::default_delete<_Tp>` Struct Template Reference

```
#include <memory>
```

#### Public Member Functions

- constexpr `default_delete` () noexcept=default
- template<typename \_Up, typename = \_Require<is\_convertible<\_Up\*, \_Tp\*>>>> constexpr `default_delete` (const `default_delete<_Up>` &) noexcept
- constexpr void `operator()` (\_Tp \*\_\_ptr) const

#### 6.385.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp>
```

Primary template of `default_delete`, used by `unique_ptr` for single objects

Since

C++11

### 6.385.2 Constructor & Destructor Documentation

#### default\_delete() [1/2]

```
template<typename _Tp>
std::default_delete< _Tp >::default_delete () [constexpr], [default], [noexcept]
Default constructor.
Referenced by default_delete\(\).
```

#### default\_delete() [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
std::default_delete< _Tp >::default_delete (
 const default_delete< _Up > &) [inline], [constexpr], [noexcept]
Converting constructor.
Allows conversion from a deleter for objects of another type, _Up, only if _Up* is convertible to _Tp*.
References default_delete\(\).
```

### 6.385.3 Member Function Documentation

#### operator>()

```
template<typename _Tp>
void std::default_delete< _Tp >::operator() (
 _Tp * __ptr) const [inline], [constexpr]
Calls delete __ptr.
```

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

## 6.386 std::default\_delete< \_Tp[]> Struct Template Reference

```
#include <memory>
```

### Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- template<typename \_Up, typename = \_Require<is\_convertible<\_Up(\*)[], \_Tp(\*)[]>>> constexpr [default\\_delete](#) (const default\_delete< \_Up[]> &) noexcept
- template<typename \_Up> constexpr [enable\\_if](#)< is\_convertible< \_Up(\*)[], \_Tp(\*)[]>::value >::type [operator](#)() (\_Up \* \_\_ptr) const

### 6.386.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete< _Tp[]>
```

Specialization of default\_delete for arrays, used by [unique\\_ptr<T\[\]>](#)

Since

C++11

### 6.386.2 Constructor & Destructor Documentation

#### default\_delete() [1/2]

```
template<typename _Tp>
std::default_delete< _Tp[]>::default_delete () [constexpr], [default], [noexcept]
```

Default constructor.  
Referenced by [default\\_delete\(\)](#).

#### default\_delete() [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
std::default_delete< _Tp[]>::default_delete (
 const default_delete< _Up[]> &) [inline], [constexpr], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for arrays of another type, such as a const-qualified version of `_Tp`.

Conversions from types derived from `_Tp` are not allowed because it is undefined to `delete[]` an array of derived types through a pointer to the base type.

References [default\\_delete\(\)](#).

### 6.386.3 Member Function Documentation

#### operator>()()

```
template<typename _Tp>
template<typename _Up>
enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type std::default_delete< _Tp[]>::←
operator() (
 _Up * __ptr) const [inline], [constexpr]
```

Calls `delete[] __ptr`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

### 6.387 \_\_gnu\_pbds::detail::default\_eq\_fn< Key > Struct Template Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef [std::equal\\_to< Key > type](#)

#### 6.387.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_eq_fn< Key >
```

Primary template, `default_eq_fn`.

#### 6.387.2 Member Typedef Documentation

##### type

```
template<typename Key>
typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:



- [standard\\_policies.hpp](#)

## 6.388 `__gnu_pbds::detail::default_hash_fn< Key >` Struct Template Reference

```
#include <standard_policies.hpp>
```

### Public Types

- `typedef std::tr1::hash< Key > type`

### 6.388.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, `default_hash_fn`.

### 6.388.2 Member Typedef Documentation

#### type

```
template<typename Key>
typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 6.389 `__gnu_parallel::default_parallel_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



### Public Member Functions

- `default_parallel_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

### 6.389.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

### 6.389.2 Member Function Documentation

#### `__get_num_threads()`

`__ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]`

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

#### `set_num_threads()`

`void __gnu_parallel::parallel_tag::set_num_threads (   
__ThreadIndex __num_threads) [inline], [inherited]`

Set the desired number of threads.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.390 `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >` Struct Template Reference

`#include <standard_policies.hpp>`

### Public Types

- `typedef cond_type::__type type`

### 6.390.1 Detailed Description

`template<typename Comb_Probe_Fn>  
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >`

Primary template, `default_probe_fn`.

### 6.390.2 Member Typedef Documentation

#### `type`

`template<typename Comb_Probe_Fn>  
typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type`

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 6.391 `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >` Struct Template Reference

```
#include <standard_policies.hpp>
```

### Public Types

- typedef `hash_standard_resize_policy< size_policy_type, trigger, false, size_type >` [type](#)

#### 6.391.1 Detailed Description

```
template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
```

Primary template, default\_resize\_policy.

#### 6.391.2 Member Typedef Documentation

##### type

```
template<typename Comb_Hash_Fn>
typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_res
Comb_Hash_Fn >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 6.392 `std::default_sentinel_t` Struct Reference

```
#include <iterator_concepts.h>
```

#### 6.392.1 Detailed Description

A sentinel type that can be used to check for the end of a range.

For some iterator types the past-the-end sentinel value is independent of the underlying sequence, and a default sentinel can be used with them. For example, a `std::counted_iterator` keeps a count of how many elements remain, and so checking for the past-the-end value only requires checking if that count has reached zero. A past-the-end `std::istream_iterator` is equal to the default-constructed value, which can be easily checked.

Comparing iterators of these types to `std::default_sentinel` is a convenient way to check if the end has been reached.

Since

C++20

The documentation for this struct was generated from the following file:

- [iterator\\_concepts.h](#)

## 6.393 `__gnu_pbds::detail::default_trie_access_traits< Key >` Struct Template Reference

#### 6.393.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >
```

Primary template, default\_trie\_access\_traits.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 6.394 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >` Struct Template Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef [trie\\_string\\_access\\_traits< string\\_type > type](#)

#### 6.394.1 Detailed Description

```
template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator<
char > > >
```

Partial specialization, `default_trie_access_traits`.

#### 6.394.2 Member Typedef Documentation

##### type

```
template<typename Char, typename Char_Traits>
typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits<
std::basic_string< Char, Char_Traits, std::allocator< char > > >::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 6.395 `__gnu_pbds::detail::default_update_policy` Struct Reference

```
#include <standard_policies.hpp>
```

#### Public Types

- typedef [lu\\_move\\_to\\_front\\_policy type](#)

#### 6.395.1 Detailed Description

Default update policy.

#### 6.395.2 Member Typedef Documentation

##### type

```
typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type
Dispatched type.
```

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 6.396 `std::defer_lock_t` Struct Reference

```
#include <std_mutex.h>
```

### 6.396.1 Detailed Description

Do not acquire ownership of the mutex.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 6.397 std::\_\_debug::deque< \_Tp, \_Allocator > Class Template Reference

```
#include <deque>
```

Inheritance diagram for std::\_\_debug::deque< \_Tp, \_Allocator >:



### Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, deque > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, deque > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **deque** (\_Base\_ref \_\_x)
- template<class \_InputIterator, typename = std::RequireInputIter< \_InputIterator >>  
**deque** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator & \_\_a= \_Allocator())
- **deque** (const \_Allocator & \_\_a)
- **deque** (const deque &)=default
- **deque** (const deque & \_\_d, const \_\_type\_identity\_t< \_Allocator > & \_\_a)
- **deque** (deque &&)=default
- **deque** (deque && \_\_d, const \_\_type\_identity\_t< \_Allocator > & \_\_a)
- **deque** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type & \_\_a=allocator\_type())
- **deque** (size\_type \_\_n, const \_\_type\_identity\_t< \_Tp > & \_\_value, const \_Allocator & \_\_a= \_Allocator())
- **deque** (size\_type \_\_n, const \_Allocator & \_\_a= \_Allocator())
- const \_Base & **\_M\_base** () const noexcept

- `_Base & _M_base ()` noexcept
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- `void assign (size_type __n, const _Tp & __t)`
- `const_reference back ()` const noexcept
- `reference back ()` noexcept
- `const_iterator begin ()` const noexcept
- `iterator begin ()` noexcept
- `const_iterator cbegin ()` const noexcept
- `const_iterator cend ()` const noexcept
- `void clear ()` noexcept
- `const_reverse_iterator crbegin ()` const noexcept
- `const_reverse_iterator crend ()` const noexcept
- `template<typename... _Args>`  
`iterator emplace (const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`reference emplace_back (_Args &&... __args)`
- `template<typename... _Args>`  
`reference emplace_front (_Args &&... __args)`
- `const_iterator end ()` const noexcept
- `iterator end ()` noexcept
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator erase (const_iterator __position)`
- `const_reference front ()` const noexcept
- `reference front ()` noexcept
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert (const_iterator __position, _Tp && __x)`
- `iterator insert (const_iterator __position, const _Tp & __x)`
- `iterator insert (const_iterator __position, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const _Tp & __x)`
- `deque & operator= (const deque &)=default`
- `deque & operator= (deque &&)=default`
- `deque & operator= (initializer_list< value_type > __l)`
- `const_reference operator[] (size_type __n)` const noexcept
- `reference operator[] (size_type __n)` noexcept
- `void pop_back ()` noexcept
- `void pop_front ()` noexcept
- `void push_back (_Tp && __x)`
- `void push_back (const _Tp & __x)`
- `void push_front (_Tp && __x)`
- `void push_front (const _Tp & __x)`
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp & __c)`
- `void shrink_to_fit ()` noexcept
- `void swap (deque & __x)` noexcept(*/\*conditional\*/*)

**Protected Member Functions**

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

**Friends**

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_Safe\_iterator

**6.397.1 Detailed Description**

template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>>  
class std::\_\_debug::deque< \_Tp, \_Allocator >

Class std::deque with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/deque](#)

**6.398 std::deque< \_Tp, \_Alloc > Class Template Reference**

```
#include <std_deque.h>
```

Inheritance diagram for std::deque< \_Tp, \_Alloc >:

**Public Types**

- typedef \_Alloc **allocator\_type**
- typedef \_Base::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef ptrdiff\_t **difference\_type**
- typedef \_Base::iterator **iterator**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- `deque()` = default
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `deque(const allocator_type &__a)`
- `deque(const deque &__x)`
- `deque(const deque &__x, const __type_identity_t< allocator_type > &__a)`
- `deque(deque &&)=default`
- `deque(deque &&__x, const __type_identity_t< allocator_type > &__a)`
- `deque(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `deque(size_type __n, const allocator_type &__a=allocator_type())`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `~deque()`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(initializer_list< value_type > __l)`
- `void assign(size_type __n, const value_type &__val)`
- `reference at(size_type __n)`
- `const_reference at(size_type __n) const`
- `const_reference back()` const noexcept
- `reference back()` noexcept
- `const_iterator begin()` const noexcept
- `iterator begin()` noexcept
- `const_iterator cbegin()` const noexcept
- `const_iterator cend()` const noexcept
- `void clear()` noexcept
- `const_reverse_iterator crbegin()` const noexcept
- `const_reverse_iterator crend()` const noexcept
- `template<typename... _Args>`  
`iterator emplace(const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`reference emplace_back(_Args &&... __args)`
- `template<typename... _Args>`  
`reference emplace_front(_Args &&... __args)`
- `bool empty()` const noexcept
- `const_iterator end()` const noexcept
- `iterator end()` noexcept
- `iterator erase(const_iterator __first, const_iterator __last)`
- `iterator erase(const_iterator __position)`
- `const_reference front()` const noexcept
- `reference front()` noexcept
- `allocator_type get_allocator()` const noexcept
- `iterator insert(const_iterator __p, initializer_list< value_type > __l)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert(const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert(const_iterator __position, const value_type &__x)`
- `iterator insert(const_iterator __position, size_type __n, const value_type &__x)`
- `iterator insert(const_iterator __position, value_type &&__x)`
- `size_type max_size()` const noexcept
- `deque & operator=(const deque &__x)`
- `deque & operator=(deque &&__x) noexcept` (`_Alloc_traits::S_always_equal()`)



- `deque` & `operator=` (`initializer_list`< `value_type` > `__l`)
- `const_reference operator[]` (`size_type __n`) `const` `noexcept`
- `reference operator[]` (`size_type __n`) `noexcept`
- `void pop_back` () `noexcept`
- `void pop_front` () `noexcept`
- `void push_back` (`const value_type &__x`)
- `void push_back` (`value_type &&__x`)
- `void push_front` (`const value_type &__x`)
- `void push_front` (`value_type &&__x`)
- `const_reverse_iterator rbegin` () `const` `noexcept`
- `reverse_iterator rbegin` () `noexcept`
- `const_reverse_iterator rend` () `const` `noexcept`
- `reverse_iterator rend` () `noexcept`
- `void resize` (`size_type __new_size`)
- `void resize` (`size_type __new_size`, `const value_type &__x`)
- `void shrink_to_fit` () `noexcept`
- `size_type size` () `const` `noexcept`
- `void swap` (`deque &__x`) `noexcept`

### Protected Types

- `enum { _S_initial_map_size }`
- `typedef __gnu_cxx:: _alloc_traits`< `_Map_alloc_type` > `_Map_alloc_traits`
- `typedef _Alloc_traits::template rebind`< `_Ptr` >::other `_Map_alloc_type`
- `typedef _Alloc_traits::pointer` `_Ptr`
- `typedef _Alloc_traits::const_pointer` `_Ptr_const`

### Protected Member Functions

- `template<typename _ForwardIterator>`  
`void _M_assign_aux` (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `template<typename _InputIterator>`  
`void _M_assign_aux` (`_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `void _M_deallocate_map` (`_Map_pointer __p`, `size_t __n`) `noexcept`
- `void _M_default_append` (`size_type __n`)
- `void _M_default_initialize` ()
- `template<typename _Alloc1>`  
`void _M_destroy_data` (`iterator __first`, `iterator __last`, `const _Alloc1 &`)
- `void _M_destroy_data` (`iterator __first`, `iterator __last`, `const std::allocator<_Tp> &`)
- `void _M_destroy_data_aux` (`iterator __first`, `iterator __last`)
- `template<typename... _Args>`  
`iterator _M_emplace_aux` (`iterator __pos`, `_Args &&... __args`)
- `iterator _M_erase` (`iterator __first`, `iterator __last`)
- `iterator _M_erase` (`iterator __pos`)
- `void _M_erase_at_begin` (`iterator __pos`)
- `void _M_erase_at_end` (`iterator __pos`)
- `void _M_fill_assign` (`size_type __n`, `const value_type &__val`)
- `void _M_fill_initialize` (`const value_type &__value`)
- `void _M_fill_insert` (`iterator __pos`, `size_type __n`, `const value_type &__x`)
- `_Map_alloc_type _M_get_map_allocator` () `const` `noexcept`
- `template<typename _ForwardIterator>`  
`void _M_insert_aux` (`iterator __pos`, `_ForwardIterator __first`, `_ForwardIterator __last`, `size_type __n`)

- iterator [\\_M\\_insert\\_aux](#) (iterator \_\_pos, const value\_type &\_\_x)
  - void [\\_M\\_insert\\_aux](#) (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
  - void [\\_M\\_move\\_assign1](#) ([deque](#) && \_\_x, [false\\_type](#))
  - void [\\_M\\_move\\_assign1](#) ([deque](#) && \_\_x, [true\\_type](#)) noexcept
  - void [\\_M\\_move\\_assign2](#) ([deque](#) && \_\_x, [false\\_type](#))
  - void [\\_M\\_move\\_assign2](#) ([deque](#) && \_\_x, [true\\_type](#))
  - template<typename \_InputIterator, typename \_Sentinel>  
void [\\_M\\_range\\_append](#) (\_InputIterator \_\_first, \_Sentinel \_\_last, size\_type \_\_n)
  - void [\\_M\\_range\\_check](#) (size\_type \_\_n) const
  - template<typename \_ForwardIterator>  
void [\\_M\\_range\\_insert\\_aux](#) (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
  - template<typename \_InputIterator>  
void [\\_M\\_range\\_insert\\_aux](#) (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
  - template<typename \_InputIterator, typename \_Sentinel>  
void [\\_M\\_range\\_prepend](#) (\_InputIterator \_\_first, \_Sentinel \_\_last, size\_type \_\_n)
  - template<typename... \_Args>  
void [\\_M\\_replace\\_map](#) (\_Args &&... \_\_args)
  - bool [\\_M\\_shrink\\_to\\_fit](#) ()
- 
- template<typename \_InputIterator>  
void [\\_M\\_range\\_initialize](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
  - template<typename \_ForwardIterator>  
void [\\_M\\_range\\_initialize](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- 
- template<typename... \_Args>  
void [\\_M\\_push\\_back\\_aux](#) (\_Args &&... \_\_args)
  - template<typename... \_Args>  
void [\\_M\\_push\\_front\\_aux](#) (\_Args &&... \_\_args)
  - void [\\_M\\_pop\\_back\\_aux](#) ()
  - void [\\_M\\_pop\\_front\\_aux](#) ()
- 
- iterator [\\_M\\_reserve\\_elements\\_at\\_front](#) (size\_type \_\_n)
  - iterator [\\_M\\_reserve\\_elements\\_at\\_back](#) (size\_type \_\_n)
  - void [\\_M\\_new\\_elements\\_at\\_front](#) (size\_type \_\_new\_elements)
  - void [\\_M\\_new\\_elements\\_at\\_back](#) (size\_type \_\_new\_elements)
- 
- void [\\_M\\_reserve\\_map\\_at\\_back](#) (size\_type \_\_nodes\_to\_add=1)
  - void [\\_M\\_reserve\\_map\\_at\\_front](#) (size\_type \_\_nodes\_to\_add=1)
  - void [\\_M\\_reallocate\\_map](#) (size\_type \_\_nodes\_to\_add, bool \_\_add\_at\_front)

### Static Protected Member Functions

- static size\_t [\\_S\\_check\\_init\\_len](#) (size\_t \_\_n, const allocator\_type &\_\_a)
- static size\_type [\\_S\\_max\\_size](#) (const \_Tp\_alloc\_type &\_\_a) noexcept

### 6.398.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque< _Tp, _Alloc >
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-*nodes*. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-`Tp`. If `Tp` is very large, there will be one `Tp` element per node (i.e., an *array* of one). For non-huge `Tp`'s, node size is inversely related to `Tp` size: the larger the `Tp`, the fewer `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different `Tp`'s, to improve allocator efficiency.

Not every pointer in the `map` array will point to a node. If the initial number of elements in the deque is small, the /middle/ `map` pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the `map` grows: available `map` pointers, if any, will be on the ends. As new nodes are created, only a subset of the `map`'s pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator `i`:
  - `i.node` points to a member of the `map` array. (Yes, you read that correctly: `i.node` does not actually point to a node.) The member of the `map` array is what actually points to the node.
  - `i.first == *(i.node)` (This points to the node (first `Tp` element).)
  - `i.last == i.first + node_size`
  - `i.cur` is a pointer in the range `[i.first, i.last)`. NOTE: the implication of this is that `i.cur` is always a dereferenceable pointer, even if `i` is a past-the-end iterator.
- `Start` and `Finish` are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with `<N` elements (where `N` is the node buffer size) must have one node, a deque with `N` through `(2N-1)` elements must have two nodes, etc.
- For every node other than `start.node` and `finish.node`, every element in the node is an initialized object. If `start.cur == finish.cur`, then `[start.cur, finish.cur)` are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, `[start.cur, start.last)` and `[finish.first, finish.cur)` are initialized objects, and `[start.first, start.cur)` and `[finish.cur, finish.last)` are uninitialized storage.
- `[map, map + map_size)` is a valid, non-empty range.
- `[start.node, finish.node]` is a valid range contained within `[map, map + map_size)`.

- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, `_Base`, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

## 6.398.2 Constructor & Destructor Documentation

### deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque () [default]
```

Creates a deque with no elements.

Referenced by [insert\(\)](#), and [operator=\(\)](#).

### deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (
 const allocator_type & __a) [inline], [explicit]
```

Creates a deque with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

### deque() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a deque with default constructed elements.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
| <code>__a</code> | An allocator.                               |

This constructor fills the deque with *n* default constructed elements.

### deque() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline]
```

Creates a deque with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the deque with `__n` copies of `__value`.

#### deque() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
 const deque<_Tp, _Alloc > & __x) [inline]
```

Deque copy constructor.

#### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

#### deque() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
 deque<_Tp, _Alloc > &&) [default]
```

Deque move constructor.

The newly-created deque contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified deque.

#### deque() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
 const deque<_Tp, _Alloc > & __x,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Copy constructor with alternative allocator.

#### deque() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc >::deque (
 deque<_Tp, _Alloc > && __x,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Move constructor with alternative allocator.

**deque()** [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 initializer_list< value_type > __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from an initializer list.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
| <code>__a</code> | An allocator object. |

Create a deque consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

**deque()** [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::deque< _Tp, _Alloc >::deque (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from a range.

**Parameters**

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

**~deque()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::~~deque () [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### 6.398.3 Member Function Documentation

#### **`_M_fill_initialize()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_fill_initialize (
 const value_type & __value) [protected]
```

Fills the deque with copies of value.

#### Parameters

|                      |                |
|----------------------|----------------|
| <code>__value</code> | Initial value. |
|----------------------|----------------|

#### Precondition

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

References [std::Destroy\(\)](#).

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::deque\(\)](#).

#### **`_M_new_elements_at_back()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_back (
 size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

References [\\_M\\_reserve\\_map\\_at\\_back\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_M\\_reserve\\_elements\\_at\\_back\(\)](#).

#### **`_M_new_elements_at_front()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_front (
 size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

References [\\_M\\_reserve\\_map\\_at\\_front\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_M\\_reserve\\_elements\\_at\\_front\(\)](#).

#### **`_M_pop_back_aux()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_back_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::pop\\_back\(\)](#).

#### **`_M_pop_front_aux()`**

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_front_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::pop\\_front\(\)](#).

**\_M\_push\_back\_aux()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_back_aux (
 _Args &&... __args) [protected]
```

Helper functions for push\_\* and pop\_\*.

References [\\_M\\_reserve\\_map\\_at\\_back\(\)](#), [std::forward\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_back\(\)](#).

**\_M\_push\_front\_aux()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_front_aux (
 _Args &&... __args) [protected]
```

Helper functions for push\_\* and pop\_\*.

References [\\_M\\_reserve\\_map\\_at\\_front\(\)](#), [std::forward\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_front\(\)](#).

**\_M\_range\_check()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_range_check (
 size_type __n) const [inline], [protected]
```

Safety check used only from at().

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::at\(\)](#), and [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::at\(\)](#).

**\_M\_range\_initialize() [1/2]**

```
template<typename _Tp, typename _Alloc>
template<typename _ForwardIterator>
void deque::_M_range_initialize (
 _ForwardIterator __first,
 _ForwardIterator __last,
 std::forward_iterator_tag) [protected]
```

Fills the deque with whatever is in [first,last).

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push\_back on each value from the iterator.

References [std::\\_Destroy\(\)](#), [std::advance\(\)](#), and [std::distance\(\)](#).

**\_M\_range\_initialize() [2/2]**

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator>
void deque::_M_range_initialize (
 _InputIterator __first,
```



```

 __InputIterator __last,
 std::input_iterator_tag) [protected]

```

Fills the deque with whatever is in [first,last).

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

References `clear()`, and `push_back()`.

Referenced by `std::deque<_Tp, polymorphic_allocator<_Tp>>::deque()`, and `std::deque<_Tp, polymorphic_allocator<_Tp>>::de`

#### `_M_reallocate_map()`

```

template<typename _Tp, typename _Alloc>
void deque::_M_reallocate_map (
 size_type __nodes_to_add,
 bool __add_at_front) [protected]

```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers.

(And consequently, deque iterators.)

References `std::max()`, and `max_size()`.

Referenced by `std::deque<_Tp, polymorphic_allocator<_Tp>>::_M_reserve_map_at_back()`, and `std::deque<_Tp, polymorphic_allocator<_Tp>>::`

#### `_M_reserve_elements_at_back()`

```

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::_M_reserve_elements_at_back (
 size_type __n) [inline], [protected]

```

Memory-handling helpers for the previous internal insert functions.

#### `_M_reserve_elements_at_front()`

```

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::_M_reserve_elements_at_front (
 size_type __n) [inline], [protected]

```

Memory-handling helpers for the previous internal insert functions.

#### `_M_reserve_map_at_back()`

```

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_reserve_map_at_back (
 size_type __nodes_to_add = 1) [inline], [protected]

```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers.

(And consequently, deque iterators.)

Referenced by `_M_new_elements_at_back()`, and `_M_push_back_aux()`.

#### `_M_reserve_map_at_front()`

```

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::_M_reserve_map_at_front (

```

```
size_type __nodes_to_add = 1) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Referenced by [\\_M\\_new\\_elements\\_at\\_front\(\)](#), and [\\_M\\_push\\_front\\_aux\(\)](#).

### assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::deque< _Tp, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a deque.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a deque with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

### assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
 initializer_list< value_type > __l) [inline]
```

Assigns an initializer list to a deque.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>↵</code>   | An initializer_list. |
| <code>__↵</code> |                      |
| <code>↵</code>   |                      |
| <code>__↵</code> |                      |
| <code>l</code>   |                      |

This function fills a deque with copies of the elements in the initializer\_list `__l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

### assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
 size_type __n,
 const value_type & __val) [inline]
```

Assigns a given value to a deque.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a deque with  $n$  copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

**at()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::at (
 size_type __n) [inline]
```

Provides access to the data contained in the deque.

**Parameters**

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <code>__↔</code><br><code>__n</code> | The index of the element for which data should be accessed. |
|--------------------------------------|-------------------------------------------------------------|

**Returns**

Read/write reference to data.

**Exceptions**

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

**at()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the deque.

**Parameters**

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <code>__↔</code><br><code>__n</code> | The index of the element for which data should be accessed. |
|--------------------------------------|-------------------------------------------------------------|

## Returns

Read-only (constant) reference to data.

## Exceptions

|                                   |                                          |
|-----------------------------------|------------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>__n</code> is an invalid index. |
|-----------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

### **back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

### **back()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

### **begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

### **begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::deque\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::deque](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::~deque\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::clear\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::front\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::front\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#), [std::operator<=>\(\)](#), and [operator=\(\)](#).

### **cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Referenced by [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#), [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#) and [std::deque< \\_Tp, polymorphic\\_allocator< \\_Tp > >::insert\(\)](#).

### **cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

**clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `_M_range_initialize()`.

**crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc>::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

**crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc>::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

**emplace()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
deque<_Tp, _Alloc>::iterator deque::emplace (
 const_iterator __position,
 _Args &&... __args)
```

Inserts an object in deque before specified iterator.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__args</code>     | Arguments.                                    |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location.` References `std::forward()`.

Referenced by `std::deque<_Tp, polymorphic_allocator<_Tp>>::insert()`.

**empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque<_Tp, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque<_Tp, _Alloc>::end () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

### end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::end () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Referenced by `std::deque<_Tp, polymorphic_allocator<_Tp>>::deque()`, `std::deque<_Tp, polymorphic_allocator<_Tp>>::deque(std::deque<_Tp, polymorphic_allocator<_Tp>>::~deque(), std::deque<_Tp, polymorphic_allocator<_Tp>>::back(), std::deque<_Tp, polymorphic_allocator<_Tp>>::back(), std::operator<=>(), and operator=()`.

### erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Remove a range of elements.

#### Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

#### Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range [`__first`,`__last`) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::erase (
 const_iterator __position) [inline]
```

Remove element at given position.

#### Parameters

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::front () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::front () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque<_Tp, _Alloc>::get_allocator () const [inline], [nodiscard], [noexcept]
```

Get a copy of the memory allocation object.

Referenced by [operator=\(\)](#).

**insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc>::insert (
 const_iterator __p,
 initializer_list<value_type> __l) [inline]
```

Inserts an initializer list into the deque.

**Parameters**

|                                                                     |                             |
|---------------------------------------------------------------------|-----------------------------|
| <a href="#"><math>\leftrightarrow</math></a><br><a href="#">__p</a> | An iterator into the deque. |
| <a href="#"><math>\leftrightarrow</math></a><br><a href="#">__l</a> | An initializer_list.        |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the initializer\_list [\\_\\_l](#) into the deque before the location specified by [\\_\\_p](#). This is known as *list insert*.

**insert()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque<_Tp, _Alloc>::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Inserts a range into the deque.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__first</code>    | An input iterator.                            |
| <code>__last</code>     | An input iterator.                            |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

**insert()** [3/5]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc >::iterator deque::insert (
 const_iterator __position,
 const value_type & __x)
```

Inserts given value into deque before specified iterator.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__x</code>        | Data to be inserted.                          |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

References [deque\(\)](#), [insert\(\)](#), [push\\_back\(\)](#), and [push\\_front\(\)](#).

Referenced by [insert\(\)](#).

**insert()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x) [inline]
```

Inserts a number of copies of given data into the deque.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__n</code>        | Number of elements to be inserted.            |
| <code>__x</code>        | Data to be inserted.                          |



**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts given rvalue into deque before specified iterator.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__x</code>        | Data to be inserted.             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size () const [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible deque.

Referenced by `_M_new_elements_at_back()`, `_M_new_elements_at_front()`, `_M_push_back_aux()`, `_M_push_front_aux()`, and `_M_reallocate_map()`.

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc > & deque::operator= (
 const deque< _Tp, _Alloc > & __x)
```

Deque assignment operator.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

All the elements of `x` are copied.

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

References `deque()`, `std::__addressof()`, `begin()`, `end()`, `get_allocator()`, `size()`, and `std::size()`.

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque< _Tp, _Alloc >::operator= (
 deque< _Tp, _Alloc > && __x) [inline], [noexcept]
```

Deque move assignment operator.

### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The contents of `__x` are moved into this deque (without copying, if the allocators permit it). `__x` is a valid, but unspecified deque.

### `operator=()` [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque<_Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Assigns an initializer list to a deque.

### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills a deque with copies of the elements in the initializer\_list `__l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

### `operator[]()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc >::operator[] (
 size_type __n) const [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

### `operator[]()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc >::operator[] (
 size_type __n) [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

#### Parameters

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| <code>_↔</code> | The index of the element for which data should be accessed. |
| <code>_n</code> |                                                             |

#### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

#### pop\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

#### pop\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

#### push\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
 const value_type & __x) [inline]
```

Add data to the end of the deque.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | Data to be added. |
| <code>_x</code> |                   |

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Referenced by `_M_range_initialize()`, and `insert()`.

#### push\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (
 const value_type & __x) [inline]
```

Add data to the front of the deque.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | Data to be added. |
| <code>_X</code> |                   |

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Referenced by [insert\(\)](#).

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

**rend()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

**rend()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
 size_type __new_size) [inline]
```

Resizes the deque to the specified number of elements.

**Parameters**

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain. |
|-------------------------|----------------------------------------------|

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
```

```
size_type __new_size,
const value_type & __x) [inline]
```

Resizes the deque to the specified number of elements.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain.      |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

#### shrink\_to\_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::shrink_to_fit () [inline], [noexcept]
```

A non-binding request to reduce memory use.

#### size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque<_Tp, _Alloc>::size () const [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the deque.

Referenced by [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::M\\_range\\_check\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::resize\(\)](#), and [std::deque<\\_Tp, polymorphic\\_allocator<\\_Tp>>::resize\(\)](#).

#### swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::swap (
 deque<_Tp, _Alloc> & __x) [inline], [noexcept]
```

Swaps data with another deque.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A deque of the same element and allocator types. |
|------------------|--------------------------------------------------|

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

## 6.399 std::destroying\_delete\_t Struct Reference

```
#include <new>
```

### 6.399.1 Detailed Description

Tag type used to declare a class-specific operator delete that can invoke the destructor before deallocating the memory.

The documentation for this struct was generated from the following file:

- [new](#)

## 6.400 `std::tr2::direct_bases<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- typedef [\\_\\_reflection\\_typelist](#)< \_\_direct\_bases(\_Tp)... > **type**

#### 6.400.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::direct_bases<_Tp>
```

Enumerate all the direct base classes of a class. Form of a typelist.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 6.401 `__gnu_pbds::direct_mask_range_hashing<Size_Type>` Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing<Size_Type>`:



### Public Types

- typedef `Size_Type` **size\_type**

### Public Member Functions

- void **swap** ([direct\\_mask\\_range\\_hashing](#)< `Size_Type` > &other)

**Protected Member Functions**

- void **notify\_resized** (size\_type size)
- size\_type **operator()** (size\_type hash) const
- size\_type **range\_hash** (size\_type hash) const
- void **swap** (mask\_based\_range\_hashing &other)

**6.401.1 Detailed Description**

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

**6.401.2 Member Function Documentation****operator()()**

```
template<typename Size_Type>
direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() (
 size_type hash) const [inline], [protected]
```

Transforms the `__hash` value hash into a ranged-hash value (using a bit-mask).

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

**6.402 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference**

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:

**Public Types**

- typedef Size\_Type **size\_type**

## Public Member Functions

- void **swap** ([direct\\_mod\\_range\\_hashing](#)< Size\_Type > &other)

## Protected Member Functions

- void **notify\_resized** (size\_type s)
- void **notify\_resized** (size\_type size)
- size\_type **operator()** (size\_type hash) const
- size\_type **range\_hash** (size\_type s) const
- void **swap** (mod\_based\_range\_hashing &other)

### 6.402.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

### 6.402.2 Member Function Documentation

#### operator>()

```
template<typename Size_Type>
direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type
>::operator() (
 size_type hash) const [inline], [protected]
```

Transforms the \_\_hash value hash into a ranged-hash value (using a modulo operation).

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 6.403 std::filesystem::directory\_entry Class Reference

```
#include <filesystem>
```

## Public Member Functions

- **directory\_entry** (const directory\_entry &)=default
- **directory\_entry** (const [filesystem::path](#) &\_\_p)
- **directory\_entry** (const [filesystem::path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- **directory\_entry** (directory\_entry &&) noexcept=default
- void **assign** (const [filesystem::path](#) &\_\_p)
- void **assign** (const [filesystem::path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool **exists** () const
- bool **exists** ([error\\_code](#) &\_\_ec) const noexcept
- uintmax\_t **file\_size** () const
- uintmax\_t **file\_size** ([error\\_code](#) &\_\_ec) const noexcept
- uintmax\_t **hard\_link\_count** () const
- uintmax\_t **hard\_link\_count** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_block\_file** () const
- bool **is\_block\_file** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_character\_file** () const
- bool **is\_character\_file** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_directory** () const



- bool **is\_directory** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_fifo** () const
- bool **is\_fifo** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_other** () const
- bool **is\_other** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_regular\_file** () const
- bool **is\_regular\_file** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_socket** () const
- bool **is\_socket** ([error\\_code](#) &\_\_ec) const noexcept
- bool **is\_symlink** () const
- bool **is\_symlink** ([error\\_code](#) &\_\_ec) const noexcept
- [file\\_time\\_type](#) **last\_write\_time** () const
- [file\\_time\\_type](#) **last\_write\_time** ([error\\_code](#) &\_\_ec) const noexcept
- **operator const filesystem::path & ()** const noexcept
- strong\_ordering **operator<=>** (const directory\_entry &\_\_rhs) const noexcept
- directory\_entry & **operator=** (const directory\_entry &)=default
- directory\_entry & **operator=** (directory\_entry &&) noexcept=default
- bool **operator==** (const directory\_entry &\_\_rhs) const noexcept
- const [filesystem::path](#) & **path** () const noexcept
- void **refresh** ()
- void **refresh** ([error\\_code](#) &\_\_ec) noexcept
- void **replace\_filename** (const [filesystem::path](#) &\_\_p)
- void **replace\_filename** (const [filesystem::path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- [file\\_status](#) **status** () const
- [file\\_status](#) **status** ([error\\_code](#) &\_\_ec) const noexcept
- [file\\_status](#) **symlink\_status** () const
- [file\\_status](#) **symlink\_status** ([error\\_code](#) &\_\_ec) const noexcept

## Friends

- struct **\_Dir**
- class **directory\_iterator**
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const directory\_iterator &\_\_d)
- class **recursive\_directory\_iterator**

### 6.403.1 Detailed Description

The value type used by directory iterators.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 6.404 std::filesystem::directory\_iterator Class Reference

```
#include <filesystem>
```

## Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [input\\_iterator\\_tag](#) **iterator\_category**
- typedef const [directory\\_entry](#) \* **pointer**
- typedef const [directory\\_entry](#) & **reference**
- typedef [directory\\_entry](#) **value\_type**

## Public Member Functions

- **directory\_iterator** (const directory\_iterator &\_\_rhs)=default
- **directory\_iterator** (const [path](#) &\_\_p)
- **directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options)
- **directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options, [error\\_code](#) &\_\_ec)
- **directory\_iterator** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- **directory\_iterator** (directory\_iterator &&\_\_rhs) noexcept=default
- directory\_iterator & **increment** ([error\\_code](#) &\_\_ec)
- const [directory\\_entry](#) & **operator\*** () const noexcept
- directory\_iterator & **operator++** ()
- \_\_directory\_iterator\_proxy **operator++** (int)
- const [directory\\_entry](#) \* **operator->** () const noexcept
- directory\_iterator & **operator=** (const directory\_iterator &\_\_rhs)=default
- directory\_iterator & **operator=** (directory\_iterator &&\_\_rhs) noexcept=default
- bool **operator==** ([default\\_sentinel\\_t](#)) const noexcept

## Friends

- bool **operator==** (const directory\_iterator &\_\_lhs, const directory\_iterator &\_\_rhs) noexcept
- class **recursive\_directory\_iterator**

## Related Symbols

(Note that these are not member symbols.)

- directory\_iterator [begin](#) (directory\_iterator \_\_iter) noexcept
- directory\_iterator [end](#) (directory\_iterator) noexcept
- recursive\_directory\_iterator [begin](#) (recursive\_directory\_iterator \_\_iter) noexcept
- recursive\_directory\_iterator [end](#) (recursive\_directory\_iterator) noexcept

### 6.404.1 Detailed Description

Iterator type for traversing the entries in a single directory.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 6.405 `std::discard_block_engine<_RandomNumberEngine, __p, __r>` Class Template Reference

```
#include <random>
```

### Public Types

- `template<typename _Sseq>`  
using `_If_seed_seq`
- `typedef _RandomNumberEngine::result_type` `result_type`

### Public Member Functions

- `discard_block_engine` ()
- `discard_block_engine` (`_RandomNumberEngine` &&\_\_rng)
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`discard_block_engine` (`_Sseq` &\_\_q)
- `discard_block_engine` (`const _RandomNumberEngine` &\_\_rng)
- `discard_block_engine` (`result_type` \_\_s)
- `const _RandomNumberEngine` & `base` () `const` `noexcept`
- `void discard` (`unsigned long long` \_\_z)
- `result_type operator()` ()
- `void seed` ()
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq>` > `seed` (`_Sseq` &\_\_q)
- `void seed` (`result_type` \_\_s)

### Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

### Static Public Attributes

- `static constexpr size_t block_size`
- `static constexpr size_t used_block`

### Friends

- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits>` & `operator<<` (`std::basic_ostream<_CharT, _Traits>` &\_\_os, `const std::discard_block_engine<_RandomNumberEngine1, __p1, __r1>` &\_\_x)
- `bool operator==` (`const discard_block_engine` &\_\_lhs, `const discard_block_engine` &\_\_rhs)
- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits>` & `operator>>` (`std::basic_istream<_CharT, _Traits>` &\_\_is, `std::discard_block_engine<_RandomNumberEngine1, __p1, __r1>` &\_\_x)

#### 6.405.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine<_RandomNumberEngine, __p, __r>
```

Produces random numbers from some base engine by discarding blocks of data.

**Precondition**

$$0 \leq r \leq p$$

**Since**

C++11

**6.405.2 Member Typedef Documentation****result\_type**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

**6.405.3 Constructor & Destructor Documentation****discard\_block\_engine() [1/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine () [inline]
```

Constructs a default `discard_block_engine` engine.

The underlying engine is default constructed as well.

**discard\_block\_engine() [2/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**discard\_block\_engine() [3/5]**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a `discard_block_engine` engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

**discard\_block\_engine()** [4/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine (
 result_type __s) [inline], [explicit]
```

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

**discard\_block\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard_block_engine (
 _Sseq & __q) [inline], [explicit]
```

Generator construct a discard\_block\_engine engine.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

**6.405.4 Member Function Documentation****base()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine & std::discard_block_engine<_RandomNumberEngine, __p, __r>::base ()
const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

**discard()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::max () [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

**min()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min () [inline],
```

[static], [constexpr]

Gets the minimum value in the generated random number range.

### operator>()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
discard_block_engine< _RandomNumberEngine, __p, __r >::result_type std::discard_block_engine< _↵
RandomNumberEngine, __p, __r >::operator() ()
```

Gets the next value in the generated random number sequence.

### seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed () [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

### seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
 _Sseq & __q) [inline]
```

Reseeds the discard\_block\_engine object with the given seed sequence.

### Parameters

|                                    |                            |
|------------------------------------|----------------------------|
| <code>_↵</code><br><code>_q</code> | A seed generator function. |
|------------------------------------|----------------------------|

### seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
 result_type __s) [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

## 6.405.5 Friends And Related Symbol Documentation

### operator<<

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↵
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x) [friend]
```

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A discard_block_engine random number generator engine. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
 const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
 const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs) [friend]
```

Compares two `discard_block_engine` random number generator objects of the same type for equality.

**Parameters**

|                    |                                                                           |
|--------------------|---------------------------------------------------------------------------|
| <code>__lhs</code> | A <code>discard_block_engine</code> random number generator object.       |
| <code>__rhs</code> | Another <code>discard_block_engine</code> random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x) [friend]
```

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                    |
| <code>__x</code>  | A <code>discard_block_engine</code> random number generator engine. |

**Returns**

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.406 `std::discrete_distribution<_IntType>` Class Template Reference**

```
#include <random>
```

## Classes

- struct [param\\_type](#)

## Public Types

- typedef [\\_IntType](#) [result\\_type](#)

## Public Member Functions

- template<typename [\\_InputIterator](#)>  
**discrete\_distribution** ([\\_InputIterator](#) \_\_wbegin, [\\_InputIterator](#) \_\_wend)
- **discrete\_distribution** (const [param\\_type](#) &\_\_p)
- **discrete\_distribution** ([initializer\\_list](#)< double > \_\_wl)
- template<typename [\\_Func](#)>  
**discrete\_distribution** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, [\\_Func](#) \_\_fw)
- template<typename [\\_ForwardIterator](#), typename [\\_UniformRandomNumberGenerator](#)>  
void **generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_ForwardIterator](#), typename [\\_UniformRandomNumberGenerator](#)>  
void **generate** ([\\_ForwardIterator](#) \_\_f, [\\_ForwardIterator](#) \_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename [\\_UniformRandomNumberGenerator](#)>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, [\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename [\\_UniformRandomNumberGenerator](#)>  
[result\\_type](#) **operator()** ([\\_UniformRandomNumberGenerator](#) &\_\_urng)
- template<typename [\\_UniformRandomNumberGenerator](#)>  
[result\\_type](#) **operator()** ([\\_UniformRandomNumberGenerator](#) &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) param () const
- void param (const [param\\_type](#) &\_\_param)
- [std::vector](#)< double > [probabilities](#) () const
- void reset ()

## Friends

- template<typename [\\_IntType1](#), typename [\\_CharT](#), typename [\\_Traits](#)>  
[std::basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > & **operator<<** ([std::basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > &\_\_os, const [std::discrete\\_distribution](#)< [\\_IntType1](#) > &\_\_x)
- bool **operator==** (const [discrete\\_distribution](#) &\_\_d1, const [discrete\\_distribution](#) &\_\_d2)
- template<typename [\\_IntType1](#), typename [\\_CharT](#), typename [\\_Traits](#)>  
[std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > & **operator>>** ([std::basic\\_istream](#)< [\\_CharT](#), [\\_Traits](#) > &\_\_is, [std::discrete\\_distribution](#)< [\\_IntType1](#) > &\_\_x)

### 6.406.1 Detailed Description

```
template<typename _IntType = int>
class std::discrete_distribution< _IntType >
```

A [discrete\\_distribution](#) random number distribution.

This distribution produces random numbers  $i$ ,  $0 \leq i < n$ , distributed according to the probability mass function  $p(i|p_0, \dots, p_{n-1}) = p_i$ .



Since

C++11

### 6.406.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 6.406.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::discrete_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**probabilities()**

```
template<typename _IntType = int>
std::vector< double > std::discrete_distribution< _IntType >::probabilities () const [inline]
Returns the probabilities of the distribution.
```

**reset()**

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset () [inline]
Resets the distribution state.
```

**6.406.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::discrete_distribution< _IntType1 > & __x) [friend]
Inserts a discrete_distribution random number distribution __x into the output stream __os.
```

**Parameters**

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__os</code> | An output stream.                                   |
| <code>__x</code>  | A discrete_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _IntType = int>
bool operator== (
 const discrete_distribution< _IntType > & __d1,
 const discrete_distribution< _IntType > & __d2) [friend]
Return true if two discrete distributions have the same parameters.
```

**operator>>**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::discrete_distribution< _IntType1 > & __x) [friend]
Extracts a discrete_distribution random number distribution __x from the input stream __is.
```

**Parameters**

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <code>__is</code> | An input stream.                                        |
| <code>__x</code>  | A discrete_distribution random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.407 std::divides< \_Tp > Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::divides< _Tp >`:

**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr `_Tp` **operator()** (const `_Tp` &`__x`, const `_Tp` &`__y`) const

**6.407.1 Detailed Description**

```
template<typename _Tp>
struct std::divides< _Tp >
```

One of the [math functors](#).

## 6.407.2 Member Typedef Documentation

### first\_argument\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Tp`, `_Tp` >::`first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

### result\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Tp`, `_Tp` >::`result_type` [inherited]

`result_type` is the return type

### second\_argument\_type

typedef `_Tp` `std::binary_function`< `_Tp`, `_Tp`, `_Tp` >::`second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.408 std::divides< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::divides< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` `is_transparent`
- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`)/↔  
`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)/`std::forward`< `_Up` >(`_u`))

### 6.408.1 Detailed Description

One of the [math functors](#).

### 6.408.2 Member Typedef Documentation

#### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

#### result\_type

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

#### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.409 std::domain\_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::domain\_error:



### Public Member Functions

- **domain\_error** (const char \*)
- **domain\_error** (const domain\_error &)=default
- **domain\_error** (const [string](#) &\_\_arg)
- **domain\_error** (domain\_error &&)=default

- domain\_error & **operator=** (const domain\_error &)=default
- domain\_error & **operator=** (domain\_error &&)=default
- virtual const char \* **what** () const noexcept

#### 6.409.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

#### 6.409.2 Member Function Documentation

##### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept\\_except.h](#)

### 6.410 [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator](#)< Key, Data, \_Alloc > Struct Template Reference

```
#include <null_node_metadata.hpp>
```

#### Public Types

- typedef const\_iterator **const\_reference**
- typedef const\_reference **reference**
- typedef const\_iterator **value\_type**

#### 6.410.1 Detailed Description

```
template<typename Key, typename Data, typename _Alloc>
```

```
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >
```

Constant node iterator.

The documentation for this struct was generated from the following file:

- [null\\_node\\_metadata.hpp](#)

### 6.411 [std::chrono::duration](#)< \_Rep, \_Period > Class Template Reference

#### Public Types

- using **period**
- using **rep**

#### Public Member Functions

- template<typename \_Rep2, typename = \_Require< is\_convertible<const \_Rep2&, rep>, \_\_or\_<\_\_is\_float<rep>, \_\_not\_<\_\_is\_float<\_\_←  
\_Rep2>>>>>>  
constexpr **duration** (const \_Rep2 &\_\_rep)
- **duration** (const duration &)=default

- template<typename \_Rep2, typename \_Period2, typename = \_Require< is\_convertible<const \_Rep2&, rep>, \_\_or\_<\_\_is\_float<rep>, \_\_and\_<\_\_is\_harmonic<\_Period2>, \_\_not\_<\_\_is\_float<\_Rep2>>>>>>  
constexpr **duration** (const duration<\_Rep2, \_Period2> &\_\_d)
- constexpr rep **count** () const
- template<typename \_Rep2 = rep>  
constexpr \_\_enable\_if\_t<![ltreat\\_as\\_floating\\_point](#)<\_Rep2>::value, duration &> **operator%=>** (const duration &\_\_d)
- template<typename \_Rep2 = rep>  
constexpr \_\_enable\_if\_t<![ltreat\\_as\\_floating\\_point](#)<\_Rep2>::value, duration &> **operator%=>** (const rep &\_\_lhs, const duration &\_\_rhs)
- constexpr duration & **operator\*=>** (const rep &\_\_rhs)
- constexpr duration< typename [common\\_type](#)< rep >::type, period > **operator+>** () const
- constexpr duration & **operator++>** ()
- constexpr duration **operator++>** (int)
- constexpr duration & **operator+=>** (const duration &\_\_d)
- constexpr duration< typename [common\\_type](#)< rep >::type, period > **operator->** () const
- constexpr duration & **operator-->** ()
- constexpr duration **operator-->** (int)
- constexpr duration & **operator-=>** (const duration &\_\_d)
- constexpr duration & **operator/=>** (const rep &\_\_rhs)
- duration & **operator=>** (const duration &)=default

### Static Public Member Functions

- static constexpr duration **max** () noexcept
- static constexpr duration **min** () noexcept
- static constexpr duration **zero** () noexcept

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr [common\\_type](#)< duration<\_Rep1, \_Period1>, duration<\_Rep2, \_Period2>>::type **operator+>** (const duration<\_Rep1, \_Period1> &\_\_lhs, const duration<\_Rep2, \_Period2> &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2>  
constexpr duration< \_\_common\_rep\_t<\_Rep1, \_Rep2>, \_Period> **operator\*>** (const duration<\_Rep1, \_Period> &\_\_d, const \_Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2>  
constexpr bool **operator==>** (const duration<\_Rep1, \_Period1> &\_\_lhs, const duration<\_Rep2, \_Period2> &\_\_rhs)

#### 6.411.1 Detailed Description

template<typename \_Rep, typename \_Period>

class std::chrono::duration<\_Rep, \_Period>

chrono::duration represents a distance between two points in time

The documentation for this class was generated from the following file:

- [chrono.h](#)

## 6.412 std::chrono::duration\_values< \_Rep > Struct Template Reference

```
#include <chrono.h>
```

### Static Public Member Functions

- static constexpr \_Rep **max** () noexcept
- static constexpr \_Rep **min** () noexcept
- static constexpr \_Rep **zero** () noexcept

#### 6.412.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::duration_values< _Rep >
```

duration\_values

The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 6.413 std::tr2::dynamic\_bitset< \_WordT, \_Alloc > Class Template Reference

```
#include <dynamic_bitset>
```

Inheritance diagram for std::tr2::dynamic\_bitset< \_WordT, \_Alloc >:



### Classes

- class [reference](#)

### Public Types

- typedef \_\_dynamic\_bitset\_base< \_WordT, \_Alloc > **\_Base**
- typedef \_Alloc **allocator\_type**
- typedef \_WordT **block\_type**
- typedef bool **const\_reference**
- typedef size\_t **size\_type**



## Public Member Functions

- [dynamic\\_bitset](#) ()=default
- [dynamic\\_bitset](#) (const allocator\_type &\_\_alloc)
- [dynamic\\_bitset](#) (const char \* \_\_str, const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (const dynamic\_bitset &)=default
- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
[dynamic\\_bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_pos=0, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_n=std::basic\_string< \_CharT, \_Traits, \_Alloc1 >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'), const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (dynamic\_bitset &&\_\_b) noexcept
- **dynamic\_bitset** ([initializer\\_list](#)< block\_type > \_\_il, const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- template<typename \_Traits = std::char\_traits<char>, typename \_CharT = typename \_Traits::char\_type>  
void **\_M\_copy\_from\_ptr** (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void **\_M\_copy\_from\_string** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1>  
void **\_M\_copy\_to\_string** ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- bool [all](#) () const
- bool [any](#) () const
- template<typename \_BlockInputIterator>  
void [append](#) (\_BlockInputIterator \_\_first, \_BlockInputIterator \_\_last)
- void [append](#) (block\_type \_\_block)
- void [append](#) ([initializer\\_list](#)< block\_type > \_\_il)
- void [clear](#) ()
- size\_type [count](#) () const noexcept
- bool [empty](#) () const noexcept
- size\_type [find\\_first](#) () const
- size\_type [find\\_next](#) (size\_t \_\_prev) const
- [dynamic\\_bitset](#) & [flip](#) ()
- [dynamic\\_bitset](#) & [flip](#) (size\_type \_\_pos)
- allocator\_type [get\\_allocator](#) () const noexcept
- bool **is\_proper\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const
- bool **is\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const
- constexpr size\_type [max\\_size](#) () noexcept
- bool [none](#) () const
- size\_type [num\\_blocks](#) () const noexcept
- [dynamic\\_bitset](#) & [operator=](#) (const [dynamic\\_bitset](#) &)=default
- [dynamic\\_bitset](#) & [operator=](#) ([dynamic\\_bitset](#) &&\_\_b) noexcept([std::is\\_nothrow\\_move\\_assignable](#)< \_Base >::value)
- [dynamic\\_bitset](#) [operator~](#) () const
- void [push\\_back](#) (bool \_\_bit)
- [dynamic\\_bitset](#) & [reset](#) ()
- [dynamic\\_bitset](#) & [reset](#) (size\_type \_\_pos)
- void [resize](#) (size\_type \_\_nbits, bool \_\_value=false)
- [dynamic\\_bitset](#) & [set](#) ()

- `dynamic_bitset` & `set` (`size_type __pos`, `bool __val=true`)
- `size_type size` () `const noexcept`
- `void swap` (`dynamic_bitset &__b`) `noexcept`
- `bool test` (`size_type __pos`) `const`
- `template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>>>`  
`std::basic_string<_CharT, _Traits, _Alloc1 > to_string` (`_CharT __zero=_CharT('0')`, `_CharT __one=_CharT('1')`)  
`const`
- `unsigned long long to_ullong` () `const`
- `unsigned long to_ulong` () `const`
- `dynamic_bitset` & `operator&=` (`const dynamic_bitset &__rhs`)
- `dynamic_bitset` & `operator&=` (`dynamic_bitset &&__rhs`)
- `dynamic_bitset` & `operator|=` (`const dynamic_bitset &__rhs`)
- `dynamic_bitset` & `operator^=` (`const dynamic_bitset &__rhs`)
- `dynamic_bitset` & `operator-=` (`const dynamic_bitset &__rhs`)
- `dynamic_bitset` & `operator<<=` (`size_type __pos`)
- `dynamic_bitset` & `operator>>=` (`size_type __pos`)
- `reference operator[]` (`size_type __pos`)
- `const_reference operator[]` (`size_type __pos`) `const`
- `dynamic_bitset operator<<` (`size_type __pos`) `const`
- `dynamic_bitset operator>>` (`size_type __pos`) `const`

### Static Public Attributes

- `static const size_type bits_per_block`
- `static const size_type npos`

### Friends

- `bool operator<` (`const dynamic_bitset &__lhs`, `const dynamic_bitset &__rhs`) `noexcept`
- `bool operator==` (`const dynamic_bitset &__lhs`, `const dynamic_bitset &__rhs`) `noexcept`
- `class reference`

### 6.413.1 Detailed Description

`template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>`  
`class std::tr2::dynamic_bitset<_WordT, _Alloc >`

The `dynamic_bitset` class represents a sequence of bits.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf>

In the general unoptimized case, storage is allocated in word-sized blocks. Let `B` be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. `B - Nb` bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `dynamic_bitset` as "a simple array of bits," be aware that your mental picture is reversed: a `dynamic_bitset` behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and

the bit at index Nb-1 in the "most significant / left-hand" position. Thus, unlike other containers, a `dynamic_bitset`'s index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
 long a = 'a';
 dynamic_bitset<> b(a);

 cout << "b('a') is " << b << endl;

 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}
```

Most of the actual code isn't contained in `dynamic_bitset<>` itself, but in the base class `__dynamic_bitset_base`. The base class works with whole words, not with individual bits. This allows us to specialize `__dynamic_bitset_base` for the important special case where the `dynamic_bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `__dynamic_bitset_base` is a vector, and is indexed as such. This is carefully encapsulated.

### 6.413.2 Constructor & Destructor Documentation

#### `dynamic_bitset()` [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset () [default]
```

All bits set to zero.

Referenced by `dynamic_bitset()`, `dynamic_bitset()`, `operator&=()`, `operator&=()`, `operator-=()`, `operator<<()`, `operator<<=()`, `operator=()`, `operator=()`, `operator>>()`, `operator^=()`, `operator|=()`, `operator~()`, and `swap()`.

#### `dynamic_bitset()` [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const allocator_type & __alloc) [inline], [explicit]
```

All bits set to zero.

#### `dynamic_bitset()` [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 size_type __nbits,
 unsigned long long __val = 0ULL,
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

#### `dynamic_bitset()` [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT, typename _Traits, typename _Alloc1>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
```

```

 typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
 typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<↵
 _CharT, _Traits, _Alloc1>::npos,
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1'),
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]

```

Use a subset of a string.

#### Parameters

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters.                        |
| <code>__pos</code>   | Index of the first character in <code>__str</code> to use. |
| <code>__n</code>     | The number of characters to copy.                          |
| <code>__zero</code>  | The character to use for unset bits.                       |
| <code>__one</code>   | The character to use for set bits.                         |
| <code>__alloc</code> | An allocator.                                              |

#### Exceptions

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>__pos</code> is bigger the size of <code>__str</code> .   |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |

References `std::basic_string< _CharT, _Traits, _Alloc >::npos`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

#### dynamic\_bitset() [5/7]

```

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const char * __str,
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]

```

Construct from a string.

#### Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters. |
| <code>__alloc</code> | An allocator.                       |

#### Exceptions

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |
|------------------------------------|--------------------------------------------------------------------|

#### dynamic\_bitset() [6/7]

```

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const dynamic_bitset< _WordT, _Alloc > &) [default]

```

Copy constructor.

References [dynamic\\_bitset\(\)](#).

### dynamic\_bitset() [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move constructor.

References [dynamic\\_bitset\(\)](#), and [std::move\(\)](#).

## 6.413.3 Member Function Documentation

### all()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all () const [inline]
```

Tests whether all the bits are on.

#### Returns

True if all the bits are set.

### any()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any () const [inline]
```

Tests whether any of the bits are on.

#### Returns

True if at least one bit is set.

### append() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
 _BlockInputIterator __first,
 _BlockInputIterator __last) [inline]
```

Append an iterator range of blocks.

References [append\(\)](#).

### append() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
 block_type __block) [inline]
```

Append a block.

Referenced by [append\(\)](#).

### clear()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::clear () [inline]
```

Clear the bitset.

**count()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::count () const [inline], [noexcept]
```

Returns the number of bits which are set.

**empty()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the dynamic\_bitset is empty.

**find\_first()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first () const [inline]
```

Finds the index of the first "on" bit.

**Returns**

The index of the first bit set, or size() if not found.

**See also**

find\_next

**find\_next()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (
 size_t __prev) const [inline]
```

Finds the index of the next "on" bit after prev.

**Returns**

The index of the next bit set, or size() if not found.

**Parameters**

|                     |                           |
|---------------------|---------------------------|
| <code>__prev</code> | Where to start searching. |
|---------------------|---------------------------|

**See also**

find\_first

**flip() [1/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip () [inline]
```

Toggles every bit to its opposite value.

Referenced by [operator~\(\)](#).

**flip()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset<_WordT, _Alloc >::flip (
 size_type __pos) [inline]
```

Toggles a given bit to its opposite value.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

**get\_allocator()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset<_WordT, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Return the allocator for the bitset.

Referenced by `operator=()`.

**max\_size()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::max_size () [inline], [constexpr], [noexcept]
```

Returns the maximum size of a dynamic\_bitset object having the same type as `*this`. The real answer is `max() * bits_per_block` but is likely to overflow.

References `std::numeric_limits<_Tp >::max()`.

**none()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset<_WordT, _Alloc >::none () const [inline]
```

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

**num\_blocks()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::num_blocks () const [inline], [noexcept]
```

Returns the total number of blocks.

**operator&=()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset<_WordT, _Alloc >::operator&= (
 const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

References [dynamic\\_bitset\(\)](#).

**operator&=()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
 dynamic_bitset< _WordT, _Alloc > && __rhs) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

References [dynamic\\_bitset\(\)](#), and [std::move\(\)](#).

**operator-=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (
 const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

References [dynamic\\_bitset\(\)](#).

**operator<<()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (
 size_type __pos) const [inline]
```

Self-explanatory.

References [dynamic\\_bitset\(\)](#).

**operator<<=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
 size_type __pos) [inline]
```

Operations on `dynamic_bitsets`.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|



These should be self-explanatory.

References [dynamic\\_bitset\(\)](#).

#### **operator=()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
 const dynamic_bitset< _WordT, _Alloc > &) [default]
```

Copy assignment operator.

References [dynamic\\_bitset\(\)](#).

#### **operator=()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
 dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move assignment operator.

References [dynamic\\_bitset\(\)](#), and [get\\_allocator\(\)](#).

#### **operator>>()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
 size_type __pos) const [inline]
```

Self-explanatory.

References [dynamic\\_bitset\(\)](#).

#### **operator>>=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
 size_type __pos) [inline]
```

Operations on dynamic\_bitsets.

#### **Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

#### **operator[]()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
 size_type __pos) [inline]
```

Array-indexing support.

#### **Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index into the dynamic_bitset. |
|--------------------|--------------------------------|

**Returns**

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

**operator[]() [2/2]**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
 size_type __pos) const [inline]
```

Array-indexing support.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index into the dynamic_bitset. |
|--------------------|--------------------------------|

**Returns**

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

**operator^=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator^= (
 const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

References [dynamic\\_bitset\(\)](#).

**operator" |=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator|= (
 const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

References [dynamic\\_bitset\(\)](#).

**operator~()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ () const [inline]
```

See the no-argument flip().

References [dynamic\\_bitset\(\)](#), and [flip\(\)](#).

**push\_back()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
 bool __bit) [inline]
```

Push a bit onto the high end of the bitset.

References [size\(\)](#).

**reset()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset () [inline]
```

Sets every bit to false.

**reset()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
 size_type __pos) [inline]
```

Sets a given bit to false.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

**Exceptions**

|                                   |                                                      |
|-----------------------------------|------------------------------------------------------|
| <a href="#">std::out_of_range</a> | If <code>__pos</code> is bigger the size of the set. |
|-----------------------------------|------------------------------------------------------|

Same as writing `set (__pos, false)`.

**resize()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (
 size_type __nbits,
 bool __value = false) [inline]
```

Resize the bitset.

Referenced by [std::tr2::operator>>\(\)](#).

**set()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set () [inline]
```

Sets every bit to true.

**set()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set (
 size_type __pos,
 bool __val = true) [inline]
```

Sets a given bit to a particular value.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__pos</code> | The index of the bit.                   |
| <code>__val</code> | Either true or false, defaults to true. |

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

**size()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size () const [inline], [noexcept]
```

Returns the total number of bits.

Referenced by [std::tr2::operator>>\(\)](#), and [push\\_back\(\)](#).

**swap()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (
 dynamic_bitset< _WordT, _Alloc > & __b) [inline], [noexcept]
```

Swap with another bitset.

References [dynamic\\_bitset\(\)](#).

**test()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (
 size_type __pos) const [inline]
```

Tests the value of a bit.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__pos</code> | The index of a bit. |
|--------------------|---------------------|

**Returns**

The value at `__pos`.

**Exceptions**

|                                                |                                                      |
|------------------------------------------------|------------------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__pos</code> is bigger the size of the set. |
|------------------------------------------------|------------------------------------------------------|

**to\_string()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string<_CharT, _Traits, _Alloc1 > std::tr2::dynamic_bitset<_WordT, _Alloc >::to_↵
string (
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1')) const [inline]
```

Returns a character interpretation of the `dynamic_bitset`.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

**to\_ullong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ullong () const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                                  |                                                                   |
|--------------------------------------------------|-------------------------------------------------------------------|
| <a href="#"><code>std::overflow_error</code></a> | If there are too many bits to be represented in an unsigned long. |
|--------------------------------------------------|-------------------------------------------------------------------|

**to\_ulong()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ulong () const [inline]
```

Returns a numerical interpretation of the `dynamic_bitset`.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                                  |                                                                   |
|--------------------------------------------------|-------------------------------------------------------------------|
| <a href="#"><code>std::overflow_error</code></a> | If there are too many bits to be represented in an unsigned long. |
|--------------------------------------------------|-------------------------------------------------------------------|

The documentation for this class was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

## 6.414 `std::ranges::elements_of< _Range, _Alloc >` Struct Template Reference

```
#include <elements_of.h>
```

### Public Attributes

- `_Alloc` **allocator**
- `_Range` **range**

#### 6.414.1 Detailed Description

```
template<range _Range, typename _Alloc = allocator<byte>>
struct std::ranges::elements_of< _Range, _Alloc >
```

Since

C++23

The documentation for this struct was generated from the following file:

- `elements_of.h`

## 6.415 `std::ranges::empty_view< _Tp >` Class Template Reference

```
#include <ranges>
```

Inheritance diagram for `std::ranges::empty_view< _Tp >`:



### Public Member Functions

- `constexpr decltype(auto)` **back** ()
- `constexpr auto` **cbegin** ()
- `constexpr auto` **cend** ()
- `constexpr decltype(auto)` **front** ()
- `constexpr operator bool` () noexcept(noexcept(ranges::empty(\_M\_derived())))
- `constexpr decltype(auto)` **operator[]** (range\_difference\_t< \_Range > \_\_n)

**Static Public Member Functions**

- `static constexpr _Tp * begin () noexcept`
- `static constexpr _Tp * data () noexcept`
- `static constexpr bool empty () noexcept`
- `static constexpr _Tp * end () noexcept`
- `static constexpr size_t size () noexcept`

**6.415.1 Detailed Description**

```
template<typename _Tp>
requires is_object_v<_Tp>
class std::ranges::empty_view<_Tp>
```

A view that contains no elements.

The documentation for this class was generated from the following file:

- [ranges](#)

**6.416 `std::enable_if< bool, _Tp >` Struct Template Reference**

```
#include <type_traits>
```

**6.416.1 Detailed Description**

```
template<bool, typename _Tp = void>
struct std::enable_if< bool, _Tp >
```

Define a member typedef `type` only if a boolean constant is true.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**6.417 `std::enable_shared_from_this< _Tp >` Class Template Reference**

```
#include <memory>
```

**Public Member Functions**

- [shared\\_ptr<\\_Tp>](#) `shared_from_this ()`
- [shared\\_ptr<const \\_Tp>](#) `shared_from_this () const`

**Protected Member Functions**

- `enable_shared_from_this (const enable_shared_from_this &) noexcept`
- `enable_shared_from_this & operator= (const enable_shared_from_this &) noexcept`

**Friends**

- `const enable_shared_from_this * __enable_shared_from_this_base (const __shared_count<> &, const enable_shared_from_this *__p)`
- `template<typename, _Lock_policy>`  
`class __shared_ptr`

### 6.417.1 Detailed Description

```
template<typename _Tp>
class std::enable_shared_from_this< _Tp >
```

Base class allowing use of the member function `shared_from_this`.

Since

C++11

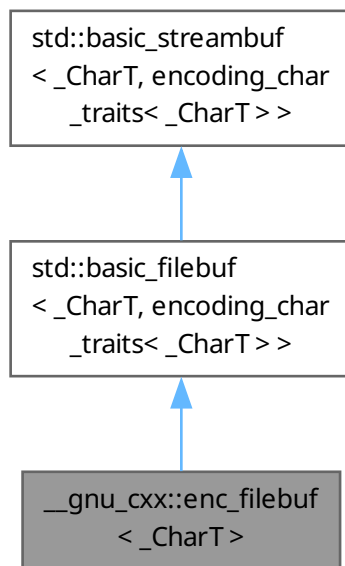
The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

### 6.418 \_\_gnu\_cxx::enc\_filebuf< \_CharT > Class Template Reference

```
#include <enc_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::enc_filebuf< _CharT >`:



#### Public Types

- typedef `codecvt< char_type, char, __state_type >` **\_\_codecvt\_type**
- typedef `__basic_file< char >` **\_\_file\_type**
- typedef `basic_filebuf< char_type, traits_type >` **filebuf\_type**
- typedef `traits_type::state_type` **\_\_state\_type**
- typedef `basic_streambuf< char_type, traits_type >` **\_\_streambuf\_type**
- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**



- typedef traits\_type::pos\_type **pos\_type**
- typedef traits\_type::state\_type **state\_type**
- typedef [encoding\\_char\\_traits](#)<\_CharT> **traits\_type**

### Public Member Functions

- **enc\_filebuf** (state\_type &\_\_state)
- `__filebuf_type` \* [close](#) ()
- locale [getloc](#) () const
- streamsize [in\\_avail](#) ()
- bool [is\\_open](#) () const throw ()
- `__filebuf_type` \* [open](#) (const char \*\_\_s, ios\_base::openmode \_\_mode)
- locale [pubimbue](#) (const locale &\_\_loc)
- int\_type [sbumpc](#) ()
- int\_type [sgetc](#) ()
- streamsize [sgetn](#) (char\_type \*\_\_s, streamsize \_\_n)
- int\_type [snextc](#) ()
- int\_type [sputbackc](#) (char\_type \_\_c)
- int\_type [sputc](#) (char\_type \_\_c)
- streamsize [sputn](#) (const char\_type \*\_\_s, streamsize \_\_n)
- int\_type [sungetc](#) ()
- void **swap** ([basic\\_filebuf](#) &)
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \*\_\_s, streamsize \_\_n)
- pos\_type [pubseekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type [pubseekpos](#) (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int [pubsync](#) ()

### Protected Member Functions

- void `__safe_gbump` (streamsize \_\_n)
- void `__safe_pbump` (streamsize \_\_n)
- void `_M_allocate_internal_buffer` ()
- bool `_M_convert_to_external` (char\_type \*, streamsize)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- int `_M_get_ext_pos` (\_\_state\_type &\_\_state)
- pos\_type `_M_seek` (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
- void `_M_set_buffer` (streamsize \_\_off)
- bool `_M_terminate_output` ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const locale &\_\_loc)
- virtual int\_type [overflow](#) (int\_type \_\_c=[encoding\\_char\\_traits](#)<\_CharT>::eof())
- virtual int\_type [pbackfail](#) (int\_type \_\_c=[encoding\\_char\\_traits](#)<\_CharT>::eof())
- void [pbump](#) (int \_\_n)
- virtual pos\_type [seekoff](#) (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- virtual pos\_type [seekpos](#) (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- virtual `__streambuf_type` \* [setbuf](#) (char\_type \*\_\_s, streamsize \_\_n)

- void [setg](#) (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void [setp](#) (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual streamsize [showmanyc](#) ()
- void [swap](#) (basic\_streambuf & \_\_sb)
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual streamsize [xsgetn](#) (char\_type \* \_\_s, streamsize \_\_n)
- virtual streamsize [xspn](#) (const char\_type \* \_\_s, streamsize \_\_n)
  
- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const
  
- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

#### Protected Attributes

- char\_type \* [\\_M\\_buf](#)
- bool [\\_M\\_buf\\_allocated](#)
- locale [\\_M\\_buf\\_locale](#)
- size\_t [\\_M\\_buf\\_size](#)
- const \_\_codecvt\_type \* [\\_M\\_codecvt](#)
- char \* [\\_M\\_ext\\_buf](#)
- streamsize [\\_M\\_ext\\_buf\\_size](#)
- char \* [\\_M\\_ext\\_end](#)
- const char \* [\\_M\\_ext\\_next](#)
- \_\_file\_type [\\_M\\_file](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- \_\_c\_lock [\\_M\\_lock](#)
- ios\_base::openmode [\\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)
- bool [\\_M\\_reading](#)
- \_\_state\_type [\\_M\\_state\\_beg](#)
- \_\_state\_type [\\_M\\_state\\_cur](#)
- \_\_state\_type [\\_M\\_state\\_last](#)
- bool [\\_M\\_writing](#)
  
- char\_type [\\_M\\_pback](#)
- char\_type \* [\\_M\\_pback\\_cur\\_save](#)
- char\_type \* [\\_M\\_pback\\_end\\_save](#)
- bool [\\_M\\_pback\\_init](#)

### 6.418.1 Detailed Description

```
template<typename _CharT>
class __gnu_cxx::enc_filebuf<_CharT>
```

class `enc_filebuf`.

### 6.418.2 Member Function Documentation

#### `_M_create_pback()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_create_pback () [inline],
[protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

#### `_M_destroy_pback()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_destroy_pback () throw ()
[inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

#### `_M_set_buffer()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer (
 streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Referenced by [seekpos\(\)](#), and [setbuf\(\)](#).

#### `close()`

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close () [inherited]
```

Closes the currently associated file.

#### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

#### `eback()`

```
char_type * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback () const [inline],
[protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

**egptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::egptr () const [inline],
[protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

**epptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::epptr () const [inline],
[protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

**gbump()**

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

**getloc()**

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::getloc () const [inline],
[inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

**gptr()**

```
char_type * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr () const [inline],
[protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

**imbue()**

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue (
 const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

**in\_avail()**

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::in_avail () [inline],
[inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**is\_open()**

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::is_open () const throw ()
[inline], [nodiscard], [inherited]
```

Returns true if the external file is open.

**open()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<↔
 _CharT, encoding_char_traits<_CharT>>::open (
 const char * __s,
 ios_base::openmode __mode) [inherited]
```

Opens an external file.

### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

### Returns

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

### overflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::overflow (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

References `_M_ext_buf`, and `_M_ext_next`.

**pbackfail()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

References `_M_ext_buf`, `_M_ext_next`, `eback()`, and `std::min()`.

**pbase()**

```
char_type * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pbase () const [inline],
[protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

**pbump()**

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

**pptr()**

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pptr () const [inline],
[protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

**pubimbue()**

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                    |         |
|--------------------|---------|
| <code>__off</code> | Offset. |
|--------------------|---------|



|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

### pubseekpos()

```
pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

### pubsetbuf()

```
basic_streambuf * std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

### pubsync()

```
int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync () [inline], [inherited]
```

Calls virtual sync function.

### sbumpc()

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sbumpc () [inline],
[inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

### seekoff()

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::pos_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::seekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.  
Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

#### seekpos()

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::pos_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::seekpos (
 pos_type __pos,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.  
Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

References `_M_reading`, and `_M_set_buffer()`.

#### setbuf()

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits<_CharT>>::setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

#### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | Pointer to a buffer area.  |
| <code>__n</code> | Size of <code>__s</code> . |

#### Returns

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

References `_M_reading`, `_M_set_buffer()`, and `gptr()`.

#### setg()

```
void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

#### Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

#### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

#### setp()

```
void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

#### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == ep_ptr()`

#### sgetc()

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetc () [inline],
[inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

#### sgetn()

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                                        |                |
|----------------------------------------|----------------|
| <a href="#"><code>_↔<br/>_s</code></a> | A buffer area. |
| <a href="#"><code>_↔<br/>_n</code></a> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc () [protected],
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**snextc()**

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::snextc () [inline],
[inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

**sputbackc()**

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputbackc (
char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                                        |                             |
|----------------------------------------|-----------------------------|
| <a href="#"><code>_↔<br/>_c</code></a> | The character to push back. |
|----------------------------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

**sputc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵__c)`.

**sputn()**

```
streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sungetc () [inline],
[inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

**sync()**

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync () [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**uflow()**

```
virtual int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::uflow () [inline], [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

**underflow()**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::underflow () [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >`.

**xsgetn()**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsgetn (char_type * __s, streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#).

**xsputn()**

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#).

**6.418.3 Member Data Documentation****`_M_buf`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf [protected],
[inherited]
```

Pointer to the beginning of internal buffer.

**`_M_buf_locale`**

```
locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale [protected],
[inherited]
```

Current locale setting.

### **`_M_buf_size`**

```
size_t std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_size [protected],
[inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

### **`_M_ext_buf`**

```
char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf [protected],
[inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Referenced by [overflow\(\)](#), and [pbackfail\(\)](#).

### **`_M_ext_buf_size`**

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected],
[inherited]
```

Size of buffer held by `_M_ext_buf`.

### **`_M_ext_next`**

```
const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected],
[inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Referenced by [overflow\(\)](#), and [pbackfail\(\)](#).

### **`_M_in_beg`**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_beg [protected],
[inherited]
```

Start of get area.

### **`_M_in_cur`**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_cur [protected],
[inherited]
```

Current read area.

### **`_M_in_end`**

```
char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_end [protected],
[inherited]
```

End of get area.

### **`_M_mode`**

```
ios_base::openmode std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_mode [protected],
[inherited]
```

Place to stash in || out || in | out settings for current filebuf.



**`_M_out_beg`**

```
char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg [protected],
[inherited]
```

Start of put area.

**`_M_out_cur`**

```
char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur [protected],
[inherited]
```

Current put area.

**`_M_out_end`**

```
char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end [protected],
[inherited]
```

End of put area.

**`_M_pback`**

```
char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback [protected],
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_cur_save`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_cur_save [protected],
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_end_save`**

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_end_save [protected],
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

**`_M_pback_init`**

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_init [protected],
[inherited]
```

Necessary bits for putback buffer management.

**Note**

pbacks of over one character are not currently supported.

## **\_M\_reading**

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading [protected],
[inherited]
```

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

Referenced by [seekpos\(\)](#), and [setbuf\(\)](#).

The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## **6.419 \_\_gnu\_cxx::encoding\_char\_traits< \_CharT > Struct Template Reference**

```
#include <codecvt_specializations.h>
```

Inheritance diagram for \_\_gnu\_cxx::encoding\_char\_traits< \_CharT >:



### **Public Types**

- typedef `_CharT` **char\_type**
- using **comparison\_category**
- typedef `_Char_types< _CharT >::int_type` **int\_type**
- typedef `_Char_types< _CharT >::off_type` **off\_type**
- typedef `std::fpos< state_type >` **pos\_type**
- typedef `encoding_state` **state\_type**

### **Static Public Member Functions**

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)

- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \*\_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 6.419.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::encoding_char_traits< _CharT >
```

encoding\_char\_traits

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

## 6.420 `__gnu_cxx::encoding_state` Class Reference

```
#include <codecvt_specializations.h>
```

### Public Types

- typedef iconv\_t **descriptor\_type**

### Public Member Functions

- **encoding\_state** (const char \*\_\_int, const char \*\_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- **encoding\_state** (const encoding\_state &\_\_obj)
- int **character\_ratio** () const
- int **external\_bom** () const
- const [std::string](#) **external\_encoding** () const
- bool **good** () const throw ()
- const descriptor\_type & **in\_descriptor** () const
- int **internal\_bom** () const
- const [std::string](#) **internal\_encoding** () const
- encoding\_state & **operator=** (const encoding\_state &\_\_obj)
- const descriptor\_type & **out\_descriptor** () const

### Protected Member Functions

- void **construct** (const encoding\_state &\_\_obj)
- void **destroy** () throw ()
- void **init** ()

### Protected Attributes

- `int _M_bytes`
- `int _M_ext_bom`
- `std::string _M_ext_enc`
- `descriptor_type _M_in_desc`
- `int _M_int_bom`
- `std::string _M_int_enc`
- `descriptor_type _M_out_desc`

#### 6.420.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

The documentation for this class was generated from the following file:

- [codecv\\_t\\_specializations.h](#)

### 6.421 `std::encoding_state` Class Reference

```
#include <codecv_t_specializations.h>
```

#### Public Types

- `typedef iconv_t descriptor_type`

#### Public Member Functions

- `encoding_state` (`const char *__int`, `const char *__ext`, `int __ibom=0`, `int __ebom=0`, `int __bytes=1`)
- `encoding_state` (`const encoding_state &__obj`)
- `int character_ratio` () `const`
- `int external_bom` () `const`
- `const std::string external_encoding` () `const`
- `bool good` () `const throw ()`
- `const descriptor_type &in_descriptor` () `const`
- `int internal_bom` () `const`
- `const std::string internal_encoding` () `const`
- `encoding_state &operator=` (`const encoding_state &__obj`)
- `const descriptor_type &out_descriptor` () `const`

#### Protected Member Functions

- `void construct` (`const encoding_state &__obj`)
- `void destroy` () `throw ()`
- `void init` ()

### Protected Attributes

- `int _M_bytes`
- `int _M_ext_bom`
- `std::string _M_ext_enc`
- `descriptor_type _M_in_desc`
- `int _M_int_bom`
- `std::string _M_int_enc`
- `descriptor_type _M_out_desc`

### 6.421.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 6.422 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>` Struct Template Reference

### 6.422.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>
```

Entry compare, primary template.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 6.423 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>` Struct Template Reference

```
#include <entry_cmp.hpp>
```

### Classes

- struct [type](#)

### Public Types

- typedef `__rebind_v::const_pointer` **entry**

### 6.423.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 6.424 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>` Struct Template Reference

```
#include <entry_cmp.hpp>
```

### Public Types

- typedef `Cmp_Fn` [type](#)

### 6.424.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true>
```

Specialization, true.

### 6.424.2 Member Typedef Documentation

#### type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
Compare.
```

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

### 6.425 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw > Struct Template Reference

#### 6.425.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >
```

Entry predicate primary class template.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

### 6.426 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false > Struct Template Reference

```
#include <entry_pred.hpp>
```

#### Public Types

- typedef \_\_rebind\_v::const\_pointer **entry**

#### 6.426.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

### 6.427 \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true > Struct Template Reference

```
#include <entry_pred.hpp>
```

#### Public Types

- typedef Pred **type**

#### 6.427.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >
```

Specialization, true.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

## 6.428 `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >` Struct Template Reference

```
#include <eq_by_less.hpp>
```

### Public Member Functions

- `bool operator() (const Key &r_lhs, const Key &r_rhs) const`

#### 6.428.1 Detailed Description

```
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >
```

Equivalence function.

The documentation for this struct was generated from the following file:

- [eq\\_by\\_less.hpp](#)

## 6.429 `__gnu_parallel::equal_split_tag` Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



#### 6.429.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

See also

`_GLIBCXX_FIND_EQUAL_SPLIT`

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.430 `__gnu_cxx::equal_to<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `__gnu_cxx::equal_to<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.430.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::equal_to<_Tp>
```

One of the [comparison functors](#).

#### 6.430.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

##### `second_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:
```



- [stl\\_function.h](#)

## 6.431 `std::equal_to<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::equal_to<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.431.1 Detailed Description

```
template<typename _Tp>
struct std::equal_to<_Tp>
```

One of the [comparison functors](#).

#### 6.431.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

## second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.432 std::ranges::equal\_to Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using **is\_transparent**

### Static Public Member Functions

- `template<typename _Tp, typename _Up>  
static constexpr bool operator() (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval<_Tp >()==std::declval<_Up >()))`

### 6.432.1 Detailed Description

ranges::equal\_to function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 6.433 std::equal\_to< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::equal\_to< void >:



### Public Types

- `typedef void first\_argument\_type`
- `typedef void first\_argument\_type`
- `typedef __is_transparent is_transparent`

- typedef bool [result\\_type](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) == [std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) == [std::forward](#)< \_Up >(\_\_u))
- constexpr bool **operator()** (const void &\_\_x, const void &\_\_y) const

#### 6.433.1 Detailed Description

One of the [comparison functors](#).

#### 6.433.2 Member Typedef Documentation

##### **first\_argument\_type** [1/2]

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### **first\_argument\_type** [2/2]

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### **result\_type** [1/2]

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type  
result\_type is the return type

##### **result\_type** [2/2]

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type  
result\_type is the return type

##### **second\_argument\_type** [1/2]

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type  
second\_argument\_type is the type of the second argument

##### **second\_argument\_type** [2/2]

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.434 std::error\_category Class Reference

```
#include <system_error>
```

## Public Member Functions

- **error\_category** (const error\_category &)=delete
- virtual [error\\_condition](#) **default\_error\_condition** (int \_\_i) const noexcept
- virtual bool [equivalent](#) (const [error\\_code](#) & \_\_code, int \_\_i) const noexcept
- virtual bool [equivalent](#) (int \_\_i, const [error\\_condition](#) & \_\_cond) const noexcept
- virtual [string](#) **message** (int) const =0
- virtual const char \* **name** () const noexcept=0
- strong\_ordering [operator<=>](#) (const error\_category & \_\_rhs) const noexcept
- error\_category & **operator=** (const error\_category &)=delete
- bool [operator==](#) (const error\_category & \_\_other) const noexcept

### 6.434.1 Detailed Description

Abstract base class for types defining a category of error codes.

An error category defines a context that gives meaning to the integer stored in an [error\\_code](#) or [error\\_condition](#) object. For example, the standard `errno` constants such as `EINVAL` and `ENOMEM` are associated with the "generic" category and other OS-specific error numbers are associated with the "system" category, but a user-defined category might give different meanings to the same numerical values.

A user-defined category can override the `equivalent` member functions to define correspondence between errors in different categories. For example, a category for errors from disk I/O could consider some of its error numbers equivalent to `ENOSPC` and `ENOENT` in the generic category.

Since

C++11

### 6.434.2 Member Function Documentation

#### **default\_error\_condition()**

```
virtual error_condition std::error_category::default_error_condition (
 int __i) const [virtual], [noexcept]
```

Return an [error\\_condition](#) corresponding to `i` in this category.

#### **equivalent()** [1/2]

```
virtual bool std::error_category::equivalent (
 const error_code & __code,
 int __i) const [virtual], [noexcept]
```

Test whether `code` corresponds to `i` for this category.

#### **equivalent()** [2/2]

```
virtual bool std::error_category::equivalent (
 int __i,
 const error_condition & __cond) const [virtual], [noexcept]
```

Test whether `cond` corresponds to `i` for this category.

#### **message()**

```
virtual string std::error_category::message (
 int) const [pure virtual]
```

A description of the error condition corresponding to the number.

References [message\(\)](#).

Referenced by [message\(\)](#), [std::error\\_code::message\(\)](#), and [std::error\\_condition::message\(\)](#).

**name()**

```
virtual const char * std::error_category::name () const [pure virtual], [noexcept]
```

A string that identifies the error category.

**operator<=>()**

```
strong_ordering std::error_category::operator<=> (
 const error_category & __rhs) const [inline], [nodiscard], [noexcept]
```

Ordered comparison that defines a total order for error categories.

References [operator<=>\(\)](#).

Referenced by [operator<=>\(\)](#).

**operator==(())**

```
bool std::error_category::operator==(
 const error_category & __other) const [inline], [noexcept]
```

An error\_category only compares equal to itself.

The documentation for this class was generated from the following file:

- [system\\_error](#)

**6.435 std::error\_code Class Reference**

```
#include <system_error>
```

**Public Member Functions**

- `template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>`  
[error\\_code](#) ([\\_ErrorCodeEnum](#) \_\_e) noexcept
- **error\_code** (const error\_code &)=default
- **error\_code** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept
- void **assign** (int \_\_v, const [error\\_category](#) &\_\_cat) noexcept
- const [error\\_category](#) & [category](#) () const noexcept
- void **clear** () noexcept
- [error\\_condition](#) [default\\_error\\_condition](#) () const noexcept
- [string](#) [message](#) () const
- [operator bool](#) () const noexcept
- error\_code & **operator=** (const error\_code &)=default
- int [value](#) () const noexcept

**Related Symbols**

(Note that these are not member symbols.)

- error\_code [make\\_error\\_code](#) (errc \_\_e) noexcept
- `template<typename _CharT, typename _Traits>`  
[basic\\_ostream](#)<\_CharT, \_Traits> & [operator<<](#) ([basic\\_ostream](#)<\_CharT, \_Traits> &\_\_os, const error\_code &\_\_e)
- strong\_ordering [operator<=>](#) (const error\_code &\_\_lhs, const error\_code &\_\_rhs) noexcept

### 6.435.1 Detailed Description

Class `error_code`

This class is a value type storing an integer error number and a category that gives meaning to the error number. Typically this is done close to the point where the error happens, to capture the original error value.

An `error_code` object can be used to store the original error value emitted by some subsystem, with a category relevant to the subsystem. For example, errors from POSIX library functions can be represented by an `errno` value and the "generic" category, but errors from an HTTP library might be represented by an HTTP response status code (e.g. 404) and a custom category defined by the library.

Since

C++11

### 6.435.2 Constructor & Destructor Documentation

#### `error_code()`

```
template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
std::error_code::error_code (
 _ErrorCodeEnum __e) [inline], [noexcept]
```

Initialize with a user-defined type, by calling `make_error_code`.

References [make\\_error\\_code\(\)](#).

### 6.435.3 Member Function Documentation

#### `category()`

```
const error_category & std::error_code::category () const [inline], [noexcept]
```

The error category that this error belongs to.

Referenced by [message\(\)](#).

#### `default_error_condition()`

```
error_condition std::error_code::default_error_condition () const [noexcept]
```

An `error_condition` for this error's category and value.

#### `message()`

```
string std::error_code::message () const [inline]
```

The category's description of the value.

References [category\(\)](#), [std::error\\_category::message\(\)](#), [message\(\)](#), and [value\(\)](#).

Referenced by [message\(\)](#).

#### `operator bool()`

```
std::error_code::operator bool () const [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

#### `value()`

```
int std::error_code::value () const [inline], [noexcept]
```

The error value.

Referenced by [message\(\)](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 6.436 `std::error_condition` Class Reference

```
#include <system_error>
```

### Public Member Functions

- `error_condition` () noexcept
- `template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>>`  
`error_condition` (\_ErrorConditionEnum \_\_e) noexcept
- `error_condition` (const `error_condition` &)=default
- `error_condition` (int \_\_v, const `error_category` &\_\_cat) noexcept
- `void assign` (int \_\_v, const `error_category` &\_\_cat) noexcept
- `const error_category & category` () const noexcept
- `void clear` () noexcept
- `string message` () const
- `operator bool` () const noexcept
- `error_condition & operator=` (const `error_condition` &)=default
- `int value` () const noexcept

### Related Symbols

(Note that these are not member symbols.)

- `error_condition make_error_condition` (errc \_\_e) noexcept
- `strong_ordering operator<=>` (const `error_condition` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- `bool operator==` (const `error_code` &\_\_lhs, const `error_code` &\_\_rhs) noexcept
- `bool operator==` (const `error_code` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- `bool operator==` (const `error_condition` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept

### 6.436.1 Detailed Description

Class `error_condition`

This class represents error conditions that may be visible at an API boundary. Different `error_code` values that can occur within a library or module might map to the same `error_condition`.

An `error_condition` represents something that the program can test for, and subsequently take appropriate action.

Since

C++11

### 6.436.2 Constructor & Destructor Documentation

#### `error_condition()` [1/3]

```
std::error_condition::error_condition () [inline], [noexcept]
```

Initialize with a zero (no error) value and the generic category.

References `std::generic_category()`.

Referenced by `make_error_condition()`, `operator<=>()`, `operator==()`, and `operator==()`.

#### `error_condition()` [2/3]

```
std::error_condition::error_condition (
 int __v,
 const error_category & __cat) [inline], [noexcept]
```

Initialize with the specified value and category.

**error\_condition()** [3/3]

```
template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
std::error_condition::error_condition (
 _ErrorConditionEnum __e) [inline], [noexcept]
```

Initialize with a user-defined type, by calling `make_error_condition`.

References [make\\_error\\_condition\(\)](#).

**6.436.3 Member Function Documentation****assign()**

```
void std::error_condition::assign (
 int __v,
 const error_category & __cat) [inline], [noexcept]
```

Set the value and category.

Referenced by [clear\(\)](#).

**category()**

```
const error_category & std::error_condition::category () const [inline], [noexcept]
```

The error category that this error belongs to.

Referenced by [message\(\)](#).

**clear()**

```
void std::error_condition::clear () [inline], [noexcept]
```

Reset the value and category to the default-constructed state.

References [assign\(\)](#), and [std::generic\\_category\(\)](#).

**message()**

```
string std::error_condition::message () const [inline]
```

The category's description of the value.

References [category\(\)](#), [std::error\\_category::message\(\)](#), and [value\(\)](#).

**operator bool()**

```
std::error_condition::operator bool () const [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

**value()**

```
int std::error_condition::value () const [inline], [noexcept]
```

The error value.

Referenced by [message\(\)](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

**6.437 \_\_gnu\_parallel::exact\_tag Struct Reference**

```
#include <tags.h>
```



Inheritance diagram for `__gnu_parallel::exact_tag`:



### Public Member Functions

- `exact_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

#### 6.437.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

#### 6.437.2 Member Function Documentation

##### `__get_num_threads()`

`_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline], [inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

void `__gnu_parallel::parallel_tag::set_num_threads` (  
     `_ThreadIndex` `__num_threads`) [inline], [inherited]

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.438 std::exception Class Reference

```
#include <exception.h>
```

Inheritance diagram for std::exception:



### Public Member Functions

- `_GLIBCXX26_CONSTEXPR exception (const exception &)=default`
- `_GLIBCXX26_CONSTEXPR exception (exception &&)=default`
- `_GLIBCXX26_CONSTEXPR exception & operator= (const exception &)=default`
- `_GLIBCXX26_CONSTEXPR exception & operator= (exception &&)=default`
- `virtual constexpr const char * what () const noexcept`

#### 6.438.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

### 6.438.2 Member Function Documentation

#### `what()`

```
virtual constexpr const char * std::exception::what () const [inline], [constexpr], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_exception](#), [std::bad\\_function\\_call](#), [std::bad\\_typeid](#), [std::bad\\_weak\\_ptr](#), [std::experimental::filesystem::v1::filesystem\\_error](#), [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), [std::filesystem::filesystem\\_error](#), [std::future\\_error](#), [std::ios\\_base::failure](#), [std::logic\\_error](#), and [std::runtime\\_error](#).

The documentation for this class was generated from the following file:

- [exception.h](#)

## 6.439 `std::__unspecified__::exception_ptr` Class Reference

```
#include <exception>
```

### Public Member Functions

- constexpr **exception\_ptr** (const exception\_ptr &) noexcept
- constexpr **exception\_ptr** (exception\_ptr &&\_\_o) noexcept
- constexpr **exception\_ptr** (nullptr\_t) noexcept
- const class [std::type\\_info](#) \* **\_\_cxa\_exception\_type** () const noexcept
- constexpr **operator bool** () const noexcept
- constexpr exception\_ptr & **operator=** (const exception\_ptr &) noexcept
- constexpr exception\_ptr & **operator=** (exception\_ptr &&\_\_o) noexcept
- constexpr void **swap** (exception\_ptr &) noexcept

### Friends

- constexpr friend bool **operator==** (const exception\_ptr &, const exception\_ptr &) noexcept=default
- exception\_ptr **std::current\_exception** () noexcept
- template<typename \_Ex>  
constexpr exception\_ptr **std::make\_exception\_ptr** (\_Ex) noexcept
- void **std::rethrow\_exception** (exception\_ptr)
- constexpr void **swap** (exception\_ptr &\_\_lhs, exception\_ptr &\_\_rhs)

### Related Symbols

(Note that these are not member symbols.)

- constexpr void **swap** (exception\_ptr &\_\_lhs, exception\_ptr &\_\_rhs)

### 6.439.1 Detailed Description

An opaque pointer to an arbitrary exception.

The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 6.440 `std::exception_ptr` Class Reference

```
#include <exception>
```

### Public Member Functions

- constexpr **exception\_ptr** (const exception\_ptr &) noexcept
- constexpr **exception\_ptr** (exception\_ptr &&\_\_o) noexcept
- constexpr **exception\_ptr** (nullptr\_t) noexcept
- const class [std::type\\_info](#) \* **\_\_cxa\_exception\_type** () const noexcept
- constexpr **operator bool** () const noexcept
- constexpr exception\_ptr & **operator=** (const exception\_ptr &) noexcept
- constexpr exception\_ptr & **operator=** (exception\_ptr &&\_\_o) noexcept
- constexpr void **swap** (exception\_ptr &) noexcept

### Friends

- constexpr friend bool **operator==** (const exception\_ptr &, const exception\_ptr &) noexcept=default
- exception\_ptr **std::current\_exception** () noexcept
- template<typename \_Ex>  
constexpr exception\_ptr **std::make\_exception\_ptr** (\_Ex) noexcept
- void **std::rethrow\_exception** (exception\_ptr)
- constexpr void **swap** (exception\_ptr &\_\_lhs, exception\_ptr &\_\_rhs)

### Related Symbols

(Note that these are not member symbols.)

- constexpr void **swap** (exception\_ptr &\_\_lhs, exception\_ptr &\_\_rhs)

#### 6.440.1 Detailed Description

An opaque pointer to an arbitrary exception.

The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

## 6.441 `std::exponential_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

**Public Member Functions**

- [exponential\\_distribution](#) ()
- [exponential\\_distribution](#) (\_RealType \_\_lambda)
- [exponential\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [template](#)<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType [lambda](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- [template](#)<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

**Friends**

- bool [operator==](#) (const [exponential\\_distribution](#) &\_\_d1, const [exponential\\_distribution](#) &\_\_d2)

**6.441.1 Detailed Description**

```
template<typename _RealType = double>
class std::exponential_distribution<_RealType>
```

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

**Table 2305: Distribution Statistics**

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda}$     |

Since

C++11

### 6.441.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::exponential_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.441.3 Constructor & Destructor Documentation

#### exponential\_distribution() [1/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution () [inline]
```

Constructs an exponential distribution with inverse scale parameter 1.0.

#### exponential\_distribution() [2/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution (
 _RealType __lambda) [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

### 6.441.4 Member Function Documentation

#### lambda()

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda () const [inline]
```

Returns the inverse scale parameter of the distribution.

#### max()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::exponential_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

### **param()** [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### **Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

### **reset()**

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Has no effect on exponential distributions.

## **6.441.5 Friends And Related Symbol Documentation**

### **operator==**

```
template<typename _RealType = double>
bool operator== (
 const exponential_distribution< _RealType > & __d1,
 const exponential_distribution< _RealType > & __d2) [friend]
```

Return true if two exponential distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## **6.442 `std::extent< typename, _Uint >` Struct Template Reference**

```
#include <type_traits>
```

Inheritance diagram for `std::extent< typename, _UInt >`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr size\_t **value**

#### 6.442.1 Detailed Description

```
template<typename, unsigned _UInt = 0>
struct std::extent< typename, _UInt >
```

extent

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 6.443 std::extreme\_value\_distribution< \_RealType > Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)



## Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const extreme\_value\_distribution &\_\_d1, const extreme\_value\_distribution &\_\_d2)

### 6.443.1 Detailed Description

template<typename \_RealType = double>  
class std::extreme\_value\_distribution<\_RealType>

A extreme\_value\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Since

C++11

### 6.443.2 Member Typedef Documentation

#### result\_type

template<typename \_RealType = double>  
typedef \_RealType std::extreme\_value\_distribution<\_RealType>::result\_type

The type of the range of the distribution.

### 6.443.3 Member Function Documentation

#### a()

template<typename \_RealType = double>  
\_RealType std::extreme\_value\_distribution<\_RealType>::a () const [inline]

Return the *a* parameter of the distribution.

**b()**

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::b () const [inline]
```

Return the  $b$  parameter of the distribution.

**max()**

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

**min()**

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::extreme_value_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param() [2/2]**

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

#### 6.443.4 Friends And Related Symbol Documentation

##### operator==

```
template<typename _RealType = double>
bool operator== (
 const extreme_value_distribution< _RealType > & __d1,
 const extreme_value_distribution< _RealType > & __d2) [friend]
```

Return true if two extreme value distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 6.444 std::locale::facet Class Reference

```
#include <locale_classes.h>
```

Inheritance diagram for `std::locale::facet`:



### Protected Member Functions

- **facet** (const facet &)=delete
- **facet** (size\_t \_\_refs=0) throw ()

- virtual [~facet\(\)](#)
- [facet](#) & **operator=** (const [facet](#) &)=delete

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

### Friends

- class `locale`
- class `locale::_Impl`

#### 6.444.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

#### 6.444.2 Constructor & Destructor Documentation

##### facet()

```
std::locale::facet::facet (
 size_t __refs = 0) throw () [inline], [explicit], [protected]
```

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

##### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__refs</code> | The initial value for reference count. |
|---------------------|----------------------------------------|

Referenced by [std::\\_\\_cxx11::collate<\\_CharT>::collate\(\)](#), [std::\\_\\_cxx11::collate<\\_CharT>::collate\(\)](#), [std::messages<\\_CharT>::messages\(\)](#), [std::messages<\\_CharT>::messages\(\)](#), [std::money\\_get<\\_CharT, \\_Inlter>::money\\_get\(\)](#), [std::money\\_put<\\_CharT, \\_Outlter>::money\\_put\(\)](#), [std::moneypunct<\\_CharT, \\_Intl>::moneypunct\(\)](#), [std::moneypunct<\\_CharT, \\_Intl>::moneypunct\(\)](#), [std::moneypunct<\\_CharT, \\_Intl>::moneypunct\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits>>::num\\_get\(\)](#), [std::num\\_put<\\_CharT, ostreambuf\\_iterator<\\_CharT, \\_Traits>>::num\\_put\(\)](#), [std::num\\_punct<\\_CharT>::num\\_punct\(\)](#), [std::num\\_punct<\\_CharT>::num\\_punct\(\)](#), [std::num\\_punct<\\_CharT>::num\\_punct\(\)](#), [std::time\\_get<\\_CharT, \\_Inlter>::time\\_get\(\)](#), [std::time\\_put<\\_CharT, \\_Outlter>::time\\_put\(\)](#), and [~facet\(\)](#).

##### ~facet()

```
virtual std::locale::facet::~~facet () [protected], [virtual]
```

Facet destructor.

References [facet\(\)](#).

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

## 6.445 std::ios\_base::failure Class Reference

```
#include <ios_base.h>
```

Inheritance diagram for std::ios\_base::failure:



### Public Member Functions

- **failure** (const char \*, const [error\\_code](#) &=io\_errc::stream)
- **failure** (const [string](#) &, const [error\\_code](#) &)
- **failure** (const [string](#) &\_\_str)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const throw ()

### 6.445.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class ios\_base::failure

### 6.445.2 Member Function Documentation

#### what()

```
virtual const char * std::ios_base::failure::what () const throw () [virtual]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::runtime\\_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 6.446 std::filesystem::file\_status Class Reference

```
#include <filesystem>
```

### Public Member Functions

- **file\_status** (const file\_status &) noexcept=default
- **file\_status** (file\_status &&) noexcept=default
- **file\_status** (file\_type \_\_ft, perms \_\_prms=perms::unknown) noexcept
- file\_status & **operator=** (const file\_status &) noexcept=default
- file\_status & **operator=** (file\_status &&) noexcept=default
- **perms permissions** () const noexcept
- void **permissions** (perms \_\_prms) noexcept
- **file\_type type** () const noexcept
- void **type** (file\_type \_\_ft) noexcept

### Friends

- bool **operator==** (const file\_status &, const file\_status &) noexcept=default

#### 6.446.1 Detailed Description

Information about a file's type and permissions.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 6.447 std::experimental::filesystem::v1::filesystem\_error Class Reference

```
#include <fs_path.h>
```

Inheritance diagram for `std::experimental::filesystem::v1::filesystem_error`:



### Public Member Functions

- **filesystem\_error** (const [string](#) & \_\_what\_arg, const [path](#) & \_\_p1, const [path](#) & \_\_p2, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) & \_\_what\_arg, const [path](#) & \_\_p1, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) & \_\_what\_arg, [error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- const [path](#) & **path1** () const noexcept
- const [path](#) & **path2** () const noexcept
- const char \* **what** () const noexcept

#### 6.447.1 Detailed Description

Exception type thrown by the Filesystem TS library.

#### 6.447.2 Member Function Documentation

##### what()

```
const char * std::experimental::filesystem::v1::filesystem_error::what () const [inline], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor). Reimplemented from [std::runtime\\_error](#).

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)



## 6.448 std::filesystem::filesystem\_error Class Reference

```
#include <filesystem>
```

Inheritance diagram for std::filesystem::filesystem\_error:



### Public Member Functions

- **filesystem\_error** (const filesystem\_error &)=default
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, [error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- filesystem\_error & **operator=** (const filesystem\_error &)=default
- const [path](#) & **path1** () const noexcept
- const [path](#) & **path2** () const noexcept
- const char \* **what** () const noexcept

### 6.448.1 Detailed Description

Exception type thrown by the Filesystem library.

Since

C++17

## 6.448.2 Member Function Documentation

### what()

```
const char * std::filesystem::filesystem_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::runtime\\_error](#).

References [std::make\\_error\\_code\(\)](#).

The documentation for this class was generated from the following file:

- [bits/fs\\_path.h](#)

## 6.449 \_\_gnu\_parallel::find\_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::find_tag`:



### 6.449.1 Detailed Description

Base class for for `std::find()` variants.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.450 std::fisher\_f\_distribution<\_RealType > Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **fisher\_f\_distribution** (\_RealType \_\_m, \_RealType \_\_n=\_RealType(1))
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::fisher\\_f\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [fisher\\_f\\_distribution](#) &\_\_d1, const [fisher\\_f\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::fisher\\_f\\_distribution](#)< \_RealType1 > &\_\_x)

### 6.450.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::fisher\_f\_distribution<\_RealType>**

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Since

C++11

### 6.450.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::fisher_f_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.450.3 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::fisher_f_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

#### 6.450.4 Friends And Related Symbol Documentation

##### operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::fisher_f_distribution<_RealType1> & __x) [friend]
```

Inserts a fisher\_f\_distribution random number distribution \_\_x into the output stream \_\_os.

##### Parameters

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__os</code> | An output stream.                                   |
| <code>__x</code>  | A fisher_f_distribution random number distribution. |

##### Returns

The output stream with the state of \_\_x inserted or in an error state.

##### operator==

```
template<typename _RealType = double>
bool operator== (
 const fisher_f_distribution<_RealType> & __d1,
 const fisher_f_distribution<_RealType> & __d2) [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

##### operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::fisher_f_distribution<_RealType1> & __x) [friend]
```

Extracts a fisher\_f\_distribution random number distribution \_\_x from the input stream \_\_is.

##### Parameters

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <code>__is</code> | An input stream.                                        |
| <code>__x</code>  | A fisher_f_distribution random number generator engine. |

##### Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.451 `__gnu_cxx::forced_error` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::forced_error`:



### Public Member Functions

- virtual constexpr const char \* [what](#) () const noexcept

#### 6.451.1 Detailed Description

Thrown by utilities for testing exception safety.

#### 6.451.2 Member Function Documentation

##### `what()`

```
virtual constexpr const char * std::exception::what () const [inline], [constexpr], [virtual],
[noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_exception](#), [std::bad\\_function\\_call](#), [std::bad\\_typeid](#), [std::bad\\_weak\\_ptr](#), [std::experimental::filesystem::v1::filesystem\\_error](#), [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), [std::filesystem::filesystem\\_error](#), [std::future\\_error](#), [std::ios\\_base::failure](#), [std::logic\\_error](#), and [std::runtime\\_error](#).

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.452 `std::forward_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::forward_iterator_tag`:



#### 6.452.1 Detailed Description

Forward iterators support a superset of input iterator operations.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 6.453 `std::__debug::forward_list<_Tp, _Alloc>` Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for `std::__debug::forward_list<_Tp, _Alloc>`:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_const_iterator, forward_list>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator<_Base_iterator, forward_list>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **forward\_list** (`_Base_ref __x`)
- template<typename `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>`>>  
**forward\_list** (`_InputIterator __first, _InputIterator __last, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`const allocator_type &__al`) noexcept
- **forward\_list** (`const forward_list &`)=default
- **forward\_list** (`const forward_list &__list, const allocator_type &__al`)
- **forward\_list** (`forward_list &&`)=default
- **forward\_list** (`forward_list &&__list, const allocator_type &__al`) noexcept(`std::is_nothrow_constructible<_Base, _Base, const allocator_type &>::value`)
- **forward\_list** (`size_type __n, const __type_identity_t<_Tp> &__value, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`size_type __n, const allocator_type &__al=allocator_type()`)
- **forward\_list** (`std::initializer_list<_Tp> __il, const allocator_type &__al=allocator_type()`)
- const `_Base &` **\_M\_base** () const noexcept
- `_Base &` **\_M\_base** () noexcept
- template<typename `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>`>>  
void **assign** (`_InputIterator __first, _InputIterator __last`)
- void **assign** (`size_type __n, const _Tp &__val`)
- void **assign** (`std::initializer_list<_Tp> __il`)
- const\_iterator **before\_begin** () const noexcept
- iterator **before\_begin** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cbegin** () const noexcept



- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- template<typename... \_Args>  
iterator **emplace\_after** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase\_after** (const\_iterator \_\_pos)
- iterator **erase\_after** (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const
- template<typename \_InputIterator, typename = std::\_\_RequireInputIter<\_InputIterator>>  
iterator **insert\_after** (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert\_after** (const\_iterator \_\_pos, \_Tp &&\_\_val)
- iterator **insert\_after** (const\_iterator \_\_pos, const \_Tp &\_\_val)
- iterator **insert\_after** (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- iterator **insert\_after** (const\_iterator \_\_pos, [std::initializer\\_list](#)<\_Tp> \_\_il)
- void **merge** (forward\_list &&\_\_list)
- template<typename \_Comp>  
void **merge** (forward\_list &&\_\_list, \_Comp \_\_comp)
- void **merge** (forward\_list &\_\_list)
- template<typename \_Comp>  
void **merge** (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & **operator=** (const forward\_list &)=default
- forward\_list & **operator=** (forward\_list &&)=default
- forward\_list & **operator=** ([std::initializer\\_list](#)<\_Tp> \_\_il)
- void **pop\_front** ()
- \_\_remove\_return\_type **remove** (const \_Tp &\_\_val)
- template<typename \_Pred>  
\_\_remove\_return\_type **remove\_if** (\_Pred \_\_pred)
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const value\_type &\_\_val)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void **swap** (forward\_list &\_\_list) noexcept(noexcept([declval](#)<\_Base &>().swap(\_\_list)))
- \_\_remove\_return\_type **unique** ()
- template<typename \_BinPred>  
\_\_remove\_return\_type **unique** (\_BinPred \_\_binary\_pred)

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)

### 6.453.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list< _Tp, _Alloc >
```

Class std::forward\_list with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

### 6.454 std::forward\_list< \_Tp, \_Alloc > Class Template Reference

```
#include <forward_list>
```

Inheritance diagram for std::forward\_list< \_Tp, \_Alloc >:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- [forward\\_list](#) ()=default
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
[forward\\_list](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const _Alloc &` \_\_al=`_Alloc()`)
- [forward\\_list](#) (`const _Alloc &` \_\_al) noexcept
- [forward\\_list](#) (`const forward_list &` \_\_list)
- [forward\\_list](#) (`const forward_list &` \_\_list, `const __type_identity_t<_Alloc> &` \_\_al)
- [forward\\_list](#) (`forward_list &&`)=default

- [forward\\_list](#) (forward\_list &&\_\_list, const \_\_type\_identity\_t< \_Alloc > &\_\_al) noexcept(\_Node\_alloc\_traits::\_S\_↵\_always\_equal())
- [forward\\_list](#) (size\_type \_\_n, const \_Alloc &\_\_al= \_Alloc())
- [forward\\_list](#) (size\_type \_\_n, const \_Tp &\_\_value, const \_Alloc &\_\_al= \_Alloc())
- [forward\\_list](#) (std::initializer\_list< \_Tp > \_\_il, const \_Alloc &\_\_al= \_Alloc())
- [~forward\\_list](#) () noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [assign](#) (size\_type \_\_n, const \_Tp &\_\_val)
- void [assign](#) (std::initializer\_list< \_Tp > \_\_il)
- const\_iterator [before\\_begin](#) () const noexcept
- iterator [before\\_begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- iterator [begin](#) () noexcept
- const\_iterator [cbefore\\_begin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... \_Args>  
iterator [emplace\\_after](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
reference [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- const\_iterator [end](#) () const noexcept
- iterator [end](#) () noexcept
- iterator [erase\\_after](#) (const\_iterator \_\_pos)
- iterator [erase\\_after](#) (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference [front](#) ()
- const\_reference [front](#) () const
- allocator\_type [get\\_allocator](#) () const noexcept
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator [insert\\_after](#) (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator [insert\\_after](#) (const\_iterator \_\_pos, \_Tp &&\_\_val)
- iterator [insert\\_after](#) (const\_iterator \_\_pos, const \_Tp &\_\_val)
- iterator [insert\\_after](#) (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- iterator [insert\\_after](#) (const\_iterator \_\_pos, std::initializer\_list< \_Tp > \_\_il)
- size\_type [max\\_size](#) () const noexcept
- void [merge](#) (forward\_list &&\_\_list)
- template<typename \_Comp>  
void [merge](#) (forward\_list &&\_\_list, \_Comp \_\_comp)
- void [merge](#) (forward\_list &\_\_list)
- template<typename \_Comp>  
void [merge](#) (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & [operator=](#) (const forward\_list &\_\_list)
- forward\_list & [operator=](#) (forward\_list &&\_\_list) noexcept(\_Node\_alloc\_traits::\_S\_nothrow\_move())
- forward\_list & [operator=](#) (std::initializer\_list< \_Tp > \_\_il)
- void [pop\\_front](#) ()
- void [push\\_front](#) (\_Tp &&\_\_val)
- void [push\\_front](#) (const \_Tp &\_\_val)
- \_\_remove\_return\_type [remove](#) (const \_Tp &\_\_val)

- `template<typename _Pred>`  
`__remove_return_type remove_if (_Pred __pred)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const value_type &__val)`
- `void reverse () noexcept`
- `void sort ()`
- `template<typename _Comp>`  
`void sort (_Comp __comp)`
- `void splice_after (const_iterator __pos, forward_list && __list) noexcept`
- `void splice_after (const_iterator __pos, forward_list && __list, const_iterator __i) noexcept`
- `void splice_after (const_iterator __pos, forward_list & __list) noexcept`
- `void splice_after (const_iterator __pos, forward_list & __list, const_iterator __i) noexcept`
- `void swap (forward_list & __list) noexcept`
- `__remove_return_type unique ()`
- `template<typename _BinPred>`  
`__remove_return_type unique (_BinPred __binary_pred)`
- `void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last) noexcept`
- `void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last) noexcept`

#### 6.454.1 Detailed Description

**template<typename \_Tp, typename \_Alloc = allocator<\_Tp>>**  
**class std::forward\_list< \_Tp, \_Alloc >**

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Since

C++11

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

#### 6.454.2 Constructor & Destructor Documentation

**forward\_list()** [1/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list () [default]
```

Creates a `forward_list` with no elements.

Referenced by `insert_after()`, `insert_after()`, `merge()`, `operator=()`, `remove()`, `remove_if()`, `resize()`, `resize()`, `sort()`, `splice_after()`, and `unique()`.

**forward\_list()** [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 const _Alloc & __al) [inline], [explicit], [noexcept]
```

Creates a forward\_list with no elements.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <code>__al</code> | An allocator object. |
|-------------------|----------------------|

**forward\_list()** [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 const forward_list< _Tp, _Alloc > & __list,
 const __type_identity_t< _Alloc > & __al) [inline]
```

Copy constructor with allocator argument.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to copy.  |
| <code>__al</code>   | An allocator object. |

**forward\_list()** [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 forward_list< _Tp, _Alloc > && __list,
 const __type_identity_t< _Alloc > & __al) [inline], [noexcept]
```

Move constructor with allocator argument.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to move.  |
| <code>__al</code>   | An allocator object. |

**forward\_list()** [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 size_type __n,
 const _Alloc & __al = _Alloc()) [inline], [explicit]
```

Creates a forward\_list with default constructed elements.

**Parameters**

|                   |                                             |
|-------------------|---------------------------------------------|
| <code>__n</code>  | The number of elements to initially create. |
| <code>__al</code> | An allocator object.                        |

This constructor creates the `forward_list` with `__n` default constructed elements.

**forward\_list()** [6/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 size_type __n,
 const _Tp & __value,
 const _Alloc & __al = _Alloc()) [inline]
```

Creates a `forward_list` with copies of an exemplar element.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__al</code>    | An allocator object.                        |

This constructor fills the `forward_list` with `__n` copies of `__value`.

**forward\_list()** [7/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::forward_list< _Tp, _Alloc >::forward_list (
 _InputIterator __first,
 _InputIterator __last,
 const _Alloc & __al = _Alloc()) [inline]
```

Builds a `forward_list` from a range.

**Parameters**

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__al</code>    | An allocator object. |

Create a `forward_list` consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

**forward\_list()** [8/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
```

```
const forward_list< _Tp, _Alloc > & __list) [inline]
```

The forward\_list copy constructor.

#### Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

#### forward\_list() [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 forward_list< _Tp, _Alloc > &&) [default]
```

The forward\_list move constructor.

#### Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The newly-created forward\_list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified forward\_list.

#### forward\_list() [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 std::initializer_list< _Tp > __il,
 const _Alloc & __al = _Alloc()) [inline]
```

Builds a forward\_list from an initializer\_list.

#### Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
| <code>__al</code> | An allocator object.               |

Create a forward\_list consisting of copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

#### ~forward\_list()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~~forward_list () [inline], [noexcept]
```

The forward\_list dtor.

### 6.454.3 Member Function Documentation

#### assign() [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::forward_list< _Tp, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a forward\_list.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a forward\_list with copies of the elements in the range [ `__first`, `__last` ).

Note that the assignment completely changes the forward\_list and that the number of elements of the resulting forward\_list is the same as the number of elements assigned.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `operator=()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::operator=()` and `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::operator=()`.

#### assign() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::assign (
 size_type __n,
 const _Tp & __val) [inline]
```

Assigns a given value to a forward\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a forward\_list with `__n` copies of the given value. Note that the assignment completely changes the forward\_list, and that the resulting forward\_list has `__n` elements.

#### assign() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::assign (
 std::initializer_list<_Tp> __il) [inline]
```

Assigns an initializer\_list to a forward\_list.

#### Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
|-------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

#### before\_begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::before_begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.



**before\_begin()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::before_begin\(\) [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\\_after\(\)](#), [insert\\_after\(\)](#), [resize\(\)](#), and [resize\(\)](#).

**begin()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::begin\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

**begin()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::begin\(\) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_back\(\)](#), and [unique\(\)](#).

**cbegin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cbegin\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::emplace\\_front\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::push\\_back\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), and [unique\(\)](#).

**cbegin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cbegin\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by [std::operator==\(const\\_iterator, const\\_iterator\)](#).

**cend()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::cend\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by [std::operator==\(const\\_iterator, const\\_iterator\)](#).

**clear()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::clear\(\) [inline], [noexcept]
```

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::operator=\(\)](#), and [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::operator=\(\)](#).

### **emplace\_after()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list<_Tp, _Alloc >::emplace_after (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Constructs object in `forward_list` after the specified iterator.

#### **Parameters**

|                     |                                                                    |
|---------------------|--------------------------------------------------------------------|
| <code>__pos</code>  | A <code>const_iterator</code> into the <code>forward_list</code> . |
| <code>__args</code> | Arguments.                                                         |

#### **Returns**

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args) ...)` after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

### **emplace\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
reference std::forward_list<_Tp, _Alloc >::emplace_front (
 _Args &&... __args) [inline]
```

Constructs object in `forward_list` at the front of the list.

#### **Parameters**

|                     |            |
|---------------------|------------|
| <code>__args</code> | Arguments. |
|---------------------|------------|

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args) ...)` at the front of the list. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

### **empty()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list<_Tp, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the `forward_list` is empty. (Thus `begin()` would equal `end()`.)

Referenced by [insert\\_after\(\)](#), and [sort\(\)](#).

### **end()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
```

```
const_iterator std::forward_list<_Tp, _Alloc>::end() const [inline], [noexcept]
```

Returns a read-only iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

#### **end()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
```

```
iterator std::forward_list<_Tp, _Alloc>::end() [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::insert_after()`, `insert_after()`, `resize()`, `resize()`, and `unique()`.

#### **erase\_after()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
```

```
iterator std::forward_list<_Tp, _Alloc>::erase_after (
 const_iterator __pos) [inline]
```

Removes the element pointed to by the iterator following `pos`.

#### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Iterator pointing before element to be erased. |
|--------------------|------------------------------------------------|

#### Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::assign()`, `std::forward_list<_Tp, polymorphic_allocator<_Tp>>::resize()`, and `resize()`.

#### **erase\_after()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
```

```
iterator std::forward_list<_Tp, _Alloc>::erase_after (
 const_iterator __pos,
 const_iterator __last) [inline]
```

Remove a range of elements.

#### Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator pointing before the first element to be erased.     |
| <code>__last</code> | Iterator pointing to one past the last element to be erased. |

**Returns**`__last`

This function will erase the elements in the range (`__pos`, `__last`) and shorten the `forward_list` accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed.

The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list< _Tp, _Alloc >::front () [inline]
```

Returns a read/write reference to the data at the first element of the `forward_list`.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front()`.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list< _Tp, _Alloc >::front () const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the `forward_list`.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Referenced by `insert_after()`, `insert_after()`, `remove()`, `remove_if()`, and `unique()`.

**insert\_after()** [1/4]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
 const_iterator __pos,
 _InputIterator __first,
 _InputIterator __last)
```

Inserts a range into the `forward_list`.

**Parameters**

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <code>__pos</code>   | An iterator into the <code>forward_list</code> . |
| <code>__first</code> | An input iterator.                               |
| <code>__last</code>  | An input iterator.                               |

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range [`__first`, `__last`) into the `forward_list` after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

References `forward_list()`, `before_begin()`, `empty()`, `end()`, and `get_allocator()`.

**insert\_after()** [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc >::insert_after (
 const_iterator __pos,
 const _Tp & __val) [inline]
```

Inserts given value into `forward_list` after specified iterator.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__val</code> | Data to be inserted.                             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::assign\(\)](#), [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\\_after\(\)](#), and [resize\(\)](#).

**insert\_after()** [3/4]

```
template<typename _Tp, typename _Alloc>
forward_list<_Tp, _Alloc >::iterator forward_list::insert_after (
 const_iterator __pos,
 size_type __n,
 const _Tp & __val)
```

Inserts a number of copies of given data into the `forward_list`.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__n</code>   | Number of elements to be inserted.               |
| <code>__val</code> | Data to be inserted.                             |

**Returns**

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`. This operation is linear in the number of elements inserted and does not invalidate iterators and references. References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), and [get\\_allocator\(\)](#).

**insert\_after()** [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc >::insert_after (
 const_iterator __pos,
 std::initializer_list<_Tp> __il) [inline]
```

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

**Parameters**

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__pos</code> | An iterator into the <code>forward_list</code> . |
| <code>__il</code>  | An initializer_list of value_type.               |

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the initializer\_list `__il` into the `forward_list` before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

**max\_size()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the largest possible number of elements of `forward_list`.

**merge() [1/2]**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::merge (
 forward_list< _Tp, _Alloc > && __list) [inline]
```

Merge sorted lists.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__list</code> | Sorted list to merge. |
|---------------------|-----------------------|

Assumes that both `__list` and this list are sorted according to `operator<()`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Referenced by `std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge()`.

**merge() [2/2]**

```
template<typename _Tp, typename _Alloc>
template<typename _Comp>
void forward_list::merge (
 forward_list< _Tp, _Alloc > && __list,
 _Comp __comp)
```

Merge sorted lists according to comparison function.

**Parameters**

|                     |                                          |
|---------------------|------------------------------------------|
| <code>__list</code> | Sorted list to merge.                    |
| <code>__comp</code> | Comparison function defining sort order. |

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

References `forward_list()`, `std::__addressof()`, and `std::move()`.

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
forward_list< _Tp, _Alloc > & forward_list::operator= (
 const forward_list< _Tp, _Alloc > & __list)
```

The forward\_list assignment operator.

**Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

All the elements of `__list` are copied.

Whether the allocator is copied depends on the allocator traits.

References [forward\\_list\(\)](#), [std::\\_\\_addressof\(\)](#), [assign\(\)](#), and [clear\(\)](#).

**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
 forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

The forward\_list move assignment operator.

**Parameters**

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The contents of `__list` are moved into this forward\_list (without copying, if the allocators permit it).

Afterwards `__list` is a valid, but unspecified forward\_list

Whether the allocator is moved depends on the allocator traits.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
 std::initializer_list< _Tp > __il) [inline]
```

The forward\_list initializer list assignment operator.

**Parameters**

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
|-------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

**pop\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the forward\_list by one. Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

**push\_front()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::push_front (
 const _Tp & __val) [inline]
```

Add data to the front of the forward\_list.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__val</code> | Data to be added. |
|--------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the forward\_list and assigns the given data to it. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

**remove()**

```
template<typename _Tp, typename _Alloc>
auto forward_list::remove (
 const _Tp & __val)
```

Remove all elements equal to value.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__val</code> | The value to remove. |
|--------------------|----------------------|

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [forward\\_list\(\)](#), [cbefore\\_begin\(\)](#), [get\\_allocator\(\)](#), and [splice\\_after\(\)](#).

**remove\_if()**

```
template<typename _Tp, typename _Alloc>
template<typename _Pred>
auto forward_list::remove_if (
 _Pred __pred) -> __remove_return_type
```

Remove all elements satisfying a predicate.

**Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__pred</code> | Unary predicate function or object. |
|---------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [forward\\_list\(\)](#), [cbefore\\_begin\(\)](#), [get\\_allocator\(\)](#), and [splice\\_after\(\)](#).

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
 size_type __sz)
```

Resizes the forward\_list to the specified number of elements.



**Parameters**

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__sz</code> | Number of elements the <code>forward_list</code> should contain. |
|-------------------|------------------------------------------------------------------|

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and the new elements are default constructed.

References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), and [erase\\_after\(\)](#).

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
 size_type __sz,
 const value_type & __val)
```

Resizes the `forward_list` to the specified number of elements.

**Parameters**

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| <code>__sz</code>  | Number of elements the <code>forward_list</code> should contain. |
| <code>__val</code> | Data with which new elements should be populated.                |

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

References [forward\\_list\(\)](#), [before\\_begin\(\)](#), [end\(\)](#), [erase\\_after\(\)](#), and [insert\\_after\(\)](#).

**reverse()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::reverse () [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

**sort()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::sort () [inline]
```

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::sort\(\)](#).

**sort()** [2/2]

```
template<typename _Tp, class _Alloc>
template<typename _Comp>
void forward_list::sort (
 _Comp __comp)
```

Sort the `forward_list` using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [forward\\_list\(\)](#), and [empty\(\)](#).

**splice\_after()** [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && ,
 const_iterator __before,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another forward\_list.

**Parameters**

|                       |                                                                         |
|-----------------------|-------------------------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.                       |
| <code>__list</code>   | Source list.                                                            |
| <code>__before</code> | Iterator referencing before the start of range in <code>__list</code> . |
| <code>__last</code>   | Iterator referencing the end of range in <code>__list</code> .          |

Removes elements in the range (`__before`, `__last`) and inserts them after `__pos` in constant time. Undefined if `__pos` is in (`__before`, `__last`).

**splice\_after()** [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

Insert contents of another forward\_list.

**Parameters**

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after. |
| <code>__list</code> | Source list.                                      |

The elements of `list` are inserted in constant time after the element referenced by `pos`. `list` becomes an empty list.

Requires `this != &x`.

Referenced by [remove\(\)](#), [remove\\_if\(\)](#), [std::forward\\_list< \\_Tp, polymorphic\\_allocator< \\_Tp > >::splice\\_after\(\)](#), and [unique\(\)](#).

**splice\_after()** [3/4]

```
template<typename _Tp, typename _Alloc>
void forward_list::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && __list,
 const_iterator __i) [noexcept]
```

Insert element from another forward\_list.

**Parameters**

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after. |
| <code>__list</code> | Source list.                                      |

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <code>__i</code> | Iterator referencing the element before the element to move. |
|------------------|--------------------------------------------------------------|

Removes the element in list `__list` referenced by `__i` and inserts it into the current list after `__pos`.  
References [forward\\_list\(\)](#).

#### **splice\_after()** [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list<_Tp, _Alloc > & ,
 const_iterator __before,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another forward\_list.

#### **Parameters**

|                       |                                                                         |
|-----------------------|-------------------------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.                       |
| <code>__list</code>   | Source list.                                                            |
| <code>__before</code> | Iterator referencing before the start of range in <code>__list</code> . |
| <code>__last</code>   | Iterator referencing the end of range in <code>__list</code> .          |

Removes elements in the range `(__before, __last)` and inserts them after `__pos` in constant time.  
Undefined if `__pos` is in `(__before, __last)`.

#### **swap()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc >::swap (
 forward_list<_Tp, _Alloc > & __list) [inline], [noexcept]
```

Swaps data with another forward\_list.

#### **Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__list</code> | A forward_list of the same element and allocator types. |
|---------------------|---------------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is overloaded such that `std::swap(l1, l2)` will feed to this function.  
Whether the allocators are swapped depends on the allocator traits.

#### **unique()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
__remove_return_type std::forward_list<_Tp, _Alloc >::unique () [inline]
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by [std::forward\\_list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::unique\(\)](#).

### unique() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinPred>
auto forward_list::unique (
 _BinPred __binary_pred) -> __remove_return_type
```

Remove consecutive elements satisfying a predicate.

#### Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>__binary_pred</code> | Binary predicate function or object. |
|----------------------------|--------------------------------------|

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [forward\\_list\(\)](#), [begin\(\)](#), [cbefore\\_begin\(\)](#), [end\(\)](#), [get\\_allocator\(\)](#), and [splice\\_after\(\)](#).

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 6.455 std::fpos<\_StateT> Class Template Reference

```
#include <postypes.h>
```

### Public Member Functions

- **fpos** (const fpos &)=default
- **fpos** (streamoff \_\_off)
- **operator streamoff** () const
- fpos **operator+** (streamoff \_\_off) const
- fpos & **operator+=** (streamoff \_\_off)
- streamoff **operator-** (const fpos &\_\_other) const
- fpos **operator-** (streamoff \_\_off) const
- fpos & **operator-=** (streamoff \_\_off)
- fpos & **operator=** (const fpos &)=default
- \_StateT **state** () const
- void **state** (\_StateT \_\_st)

#### 6.455.1 Detailed Description

```
template<typename _StateT>
class std::fpos<_StateT>
```

Class representing stream positions.

The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

**Parameters**

|               |                                           |
|---------------|-------------------------------------------|
| <i>StateT</i> | Type passed to and returned from state(). |
|---------------|-------------------------------------------|

**6.455.2 Constructor & Destructor Documentation****fpos()**

```
template<typename _StateT>
std::fpos<_StateT>::fpos (
 streamoff __off) [inline]
```

Construct position from offset.

**6.455.3 Member Function Documentation****operator streamoff()**

```
template<typename _StateT>
std::fpos<_StateT>::operator streamoff () const [inline]
```

Convert to streamoff.

**operator+()**

```
template<typename _StateT>
fpos std::fpos<_StateT>::operator+ (
 streamoff __off) const [inline]
```

Add position and offset.

**operator+=()**

```
template<typename _StateT>
fpos & std::fpos<_StateT>::operator+= (
 streamoff __off) [inline]
```

Add offset to this position.

**operator-() [1/2]**

```
template<typename _StateT>
streamoff std::fpos<_StateT>::operator- (
 const fpos<_StateT> & __other) const [inline]
```

Subtract position to return offset.

**operator-() [2/2]**

```
template<typename _StateT>
fpos std::fpos<_StateT>::operator- (
 streamoff __off) const [inline]
```

Subtract offset from position.

**operator-=()**

```
template<typename _StateT>
fpos & std::fpos<_StateT>::operator-= (
 streamoff __off) [inline]
```

Subtract offset from this position.

**state()** [1/2]

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state () const [inline]
Return the last set value of st.
```

**state()** [2/2]

```
template<typename _StateT>
void std::fpos< _StateT >::state (
 _StateT __st) [inline]
```

Remember the value of *st*.

The documentation for this class was generated from the following file:

- [postypes.h](#)

**6.456 \_\_gnu\_cxx::free\_list Class Reference**

```
#include <bitmap_allocator.h>
```

Inheritance diagram for `__gnu_cxx::free_list`:

**Public Types**

- typedef `__mutex` **mutex\_type**
- typedef `vector_type::iterator` **iterator**
- typedef `std::size_t *` **value\_type**
- typedef `__detail::__mini_vector< value_type >` **vector\_type**

**Public Member Functions**

- void `_M_clear` ()
- `std::size_t *` `_M_get` (`std::size_t` \_\_sz)
- void `_M_insert` (`std::size_t *` \_\_addr) throw ()

**6.456.1 Detailed Description**

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

### 6.456.2 Member Function Documentation

#### **`_M_clear()`**

```
void __gnu_cxx::free_list::_M_clear ()
```

This function just clears the internal Free List, and gives back all the memory to the OS.

#### **`_M_get()`**

```
std::size_t * __gnu_cxx::free_list::_M_get (
 std::size_t __sz)
```

This function gets a block of memory of the specified size from the free list.

#### Parameters

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__sz</code> | The size in bytes of the memory required. |
|-------------------|-------------------------------------------|

#### Returns

A pointer to the new memory block of size at least equal to that requested.

#### **`_M_insert()`**

```
void __gnu_cxx::free_list::_M_insert (
 std::size_t * __addr) throw () [inline]
```

This function returns the block of memory to the internal free list.

#### Parameters

|                     |                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| <code>__addr</code> | The pointer to the memory block that was given by a call to the <code>_M_get</code> function. |
|---------------------|-----------------------------------------------------------------------------------------------|

Referenced by `__gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object()`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 6.457 std::from\_chars\_result Struct Reference

```
#include <charconv>
```

### Public Member Functions

- constexpr **operator bool** () const noexcept

### Public Attributes

- `errc` **ec**
- `const char *` **ptr**

### Friends

- `bool` **operator==** (const [from\\_chars\\_result](#) &, const [from\\_chars\\_result](#) &)=default

### 6.457.1 Detailed Description

Result type of `std::from_chars`.

The documentation for this struct was generated from the following file:

- [charconv](#)

## 6.458 `std::front_insert_iterator< _Container >` Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::front_insert_iterator< _Container >`:



### Public Types

- typedef `_Container` [container\\_type](#)
- using **difference\_type**
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- constexpr [front\\_insert\\_iterator](#) (`_Container &__x`)
- constexpr [front\\_insert\\_iterator](#) & [operator\\*](#) ()
- constexpr [front\\_insert\\_iterator](#) & [operator++](#) ()
- constexpr [front\\_insert\\_iterator](#) [operator++](#) (int)
- constexpr [front\\_insert\\_iterator](#) & [operator=](#) (const typename `_Container::value_type` &\_\_value)
- constexpr [front\\_insert\\_iterator](#) & [operator=](#) (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container *` **container**



### 6.458.1 Detailed Description

```
template<typename _Container>
class std::front_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push\_front.

Tip: Using the front\_inserter function to create these iterators can save typing.

### 6.458.2 Member Typedef Documentation

#### container\_type

```
template<typename _Container>
typedef _Container std::front_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

#### iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
iterator_category [inherited]
```

One of the [tag types](#).

#### pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

#### reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

#### value\_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

### 6.458.3 Constructor & Destructor Documentation

#### front\_insert\_iterator()

```
template<typename _Container>
std::front_insert_iterator< _Container >::front_insert_iterator (
 _Container & __x) [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.

References [std::\\_\\_addressof\(\)](#).

### 6.458.4 Member Function Documentation

#### operator\*()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator* () [inline], [nodiscard],
[constexpr]
```

Simply returns \*this.

**operator++()** [1/2]

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator++ () [inline], [constexpr]
Simply returns *this. (This iterator does not move).
```

**operator++()** [2/2]

```
template<typename _Container>
front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
 int) [inline], [constexpr]
Simply returns *this. (This iterator does not move).
```

**operator=()**

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator= (
 const typename _Container::value_type & __value) [inline], [constexpr]
```

**Parameters**

|                      |                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const <code>T</code> for <code>container&lt;T&gt;</code> . |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container. The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**6.459 std::function< \_Res(\_ArgTypes...)> Class Template Reference**

```
#include <std_function.h>
```

Inheritance diagram for `std::function< _Res(_ArgTypes...)>`:



## Public Types

- typedef `_Res` **result\_type**

## Public Member Functions

- `function` () noexcept
- template<typename `_Functor`, typename `_Constraints` = `_Requires<_Callable<_Functor>>>`  
`function` (`_Functor` &&\_\_f) noexcept(`_Handler<_Functor>::template _S_nothrow_init<_Functor>()`)
- `function` (const function &\_\_x)
- `function` (function &&\_\_x) noexcept
- `function` (nullptr\_t) noexcept
- `operator bool` () const noexcept
- `_Res operator()` (`_ArgTypes...` \_\_args) const
- template<typename `_Functor`>  
`_Requires<_Callable<_Functor>, function &> operator=` (`_Functor` &&\_\_f) noexcept(`_Handler<_Functor>::template _S_nothrow_init<_Functor>()`)
- `function` & `operator=` (const `function` &\_\_x)
- `function` & `operator=` (`function` &&\_\_x) noexcept
- `function` & `operator=` (nullptr\_t) noexcept
- template<typename `_Functor`>  
`function` & `operator=` (`reference_wrapper<_Functor>` \_\_f) noexcept
- void `swap` (`function` &\_\_x) noexcept
- const `type_info` & `target_type` () const noexcept
- template<typename `_Functor`>  
`_Functor * target` () noexcept
- template<typename `_Functor`>  
const `_Functor * target` () const noexcept

### 6.459.1 Detailed Description

template<typename `_Res`, typename... `_ArgTypes`>  
class std::function< `_Res`(`_ArgTypes...`)>

Polymorphic function wrapper.

Since

C++11

### 6.459.2 Constructor & Destructor Documentation

`function()` [1/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function () [inline], [noexcept]
```

Default construct creates an empty function call wrapper.

Postcondition

```
!(bool)*this
```

References `function()`.

Referenced by `function()`, `function()`, `function()`, `function()`, `function()`, `operator=()`, `operator=()`, `operator=()`, `operator=()`, `operator=()`, `swap()`, and `target()`.

**function()** [2/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 nullptr_t) [inline], [noexcept]
```

Creates an empty function call wrapper.

**Postcondition**

!(bool)\*this

References [function\(\)](#).

**function()** [3/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function copy constructor.

**Parameters**

|                                                                                          |                                                  |
|------------------------------------------------------------------------------------------|--------------------------------------------------|
| <br>__x | A function object with identical call signature. |
|------------------------------------------------------------------------------------------|--------------------------------------------------|

**Postcondition**

bool(\*this) == bool(\_\_x)

The newly-created function contains a copy of the target of \_\_x (if it has one).

References [function\(\)](#).

**function()** [4/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move constructor.

**Parameters**

|                                                                                            |                                                         |
|--------------------------------------------------------------------------------------------|---------------------------------------------------------|
| <br>__x | A function object rvalue with identical call signature. |
|--------------------------------------------------------------------------------------------|---------------------------------------------------------|

The newly-created function contains the target of \_\_x (if it has one).

References [function\(\)](#).

**function()** [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
std::function< _Res(_ArgTypes...)>::function (
 _Functor && __f) [inline], [noexcept]
```

Builds a function that targets a copy of the incoming function object.

**Parameters**

|                |                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>↔</code> | A function object that is callable with parameters of type <code>ArgTypes...</code> and returns a value convertible to <code>Res</code> . |
| <code>↔</code> |                                                                                                                                           |
| <code>↔</code> |                                                                                                                                           |
| <code>↔</code> |                                                                                                                                           |
| <code>f</code> |                                                                                                                                           |

The newly-created function object will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer, null pointer-to-member, or empty `std::function`, the newly-created object will be empty.

If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

References [function\(\)](#), and [std::forward\(\)](#).

**6.459.3 Member Function Documentation****`operator bool()`**

```
template<typename _Res, typename... _ArgTypes>
std::function<_Res(_ArgTypes...)>::operator bool () const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw exceptions.

**`operator()()`**

```
template<typename _Res, typename... _ArgTypes>
_Res std::function<_Res(_ArgTypes...)>::operator() (
 _ArgTypes... __args) const [inline]
```

Invokes the function targeted by `*this`.

**Returns**

the result of the target.

**Exceptions**

|                                  |                                |
|----------------------------------|--------------------------------|
| <code>'bad_function_call'</code> | when <code>!(bool)*this</code> |
|----------------------------------|--------------------------------|

The function call operator invokes the target function object stored by `this`.

References [std::forward\(\)](#), and [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

**`operator=()` [1/5]**

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Requires<_Callable<_Functor>, function &> std::function<_Res(_ArgTypes...)>::operator= (
 _Functor && __f) [inline], [noexcept]
```

Function assignment to a new target.

**Parameters**

|                      |                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
| $\leftrightarrow$    | A function object that is callable with parameters of type <code>_ArgTypes...</code> and returns a value convertible to |
| $\_ \leftrightarrow$ | <code>_Res</code> .                                                                                                     |
| $\leftrightarrow$    |                                                                                                                         |
| $\_ \leftrightarrow$ |                                                                                                                         |
| <code>f</code>       |                                                                                                                         |

**Returns**

`*this`

**Since**

C++11

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer or null pointer-to-member, this object will be empty.

If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw. References [function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

**operator=()** [2/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
 const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function assignment operator.

**Parameters**

|                      |                                           |
|----------------------|-------------------------------------------|
| $\_ \leftrightarrow$ | A function with identical call signature. |
| <code>__x</code>     |                                           |

**Postcondition**

`(bool)*this == (bool)x`

**Returns**

`*this`

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

References [function\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**operator=()** [3/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
 function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move-assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A function rvalue with identical call signature. |
|------------------|--------------------------------------------------|

**Returns**

`*this`

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

References [function\(\)](#), [std::move\(\)](#), and [operator=\(\)](#).

**operator=()** [4/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
 nullptr_t) [inline], [noexcept]
```

Function assignment to empty.

**Postcondition**

`!(bool)*this`

**Returns**

`*this`

The target of `*this` is deallocated, leaving it empty.

References [function\(\)](#), and [operator=\(\)](#).

**operator=()** [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
function & std::function< _Res(_ArgTypes...)>::operator= (
 reference_wrapper< _Functor > __f) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

References [function\(\)](#), and [operator=\(\)](#).

**swap()**

```
template<typename _Res, typename... _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
 function< _Res(_ArgTypes...)> & __x) [inline], [noexcept]
```

Swap the targets of two function objects.

**Parameters**

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__x</code> | A function with identical call signature. |
|------------------|-------------------------------------------|

Swap the targets of `this` function object and `__f`. This function will not throw exceptions.

References [function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#).

**target() [1/2]**

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
const _Functor * std::function< _Res(_ArgTypes...)>::target () const [inline], [noexcept]
```

Access the stored target function object.

**Returns**

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.

References [target\(\)](#), and [target\\_type\(\)](#).

**target() [2/2]**

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Functor * std::function< _Res(_ArgTypes...)>::target () [inline], [noexcept]
```

Access the stored target function object.

**Returns**

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.

References [function\(\)](#), and [target\(\)](#).

Referenced by [target\(\)](#), and [target\(\)](#).

**target\_type()**

```
template<typename _Res, typename... _ArgTypes>
const type_info & std::function< _Res(_ArgTypes...)>::target_type () const [inline], [noexcept]
```

Determine the type of the target of this function object wrapper.

**Returns**

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw exceptions.

References [target\\_type\(\)](#).

Referenced by [target\(\)](#), and [target\\_type\(\)](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

## 6.460 `std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>` Class Template Reference

```
#include <functional>
```

**Public Member Functions**

- `template<typename _Fn, typename _Vt = remove_reference_t<_Fn>> constexpr function\_ref (_Fn &&__f) noexcept`
- `template<typename _Fn> function\_ref (_Fn *__fn) noexcept`



- `template<auto __fn, typename _Fn>`  
`constexpr function_ref (constant_wrapper< __fn, _Fn >) noexcept`
- `template<auto __fn, typename _Fn, typename _Td>`  
`constexpr function_ref (constant_wrapper< __fn, _Fn >, _Td _GLIBCXX_MOF_CV * __ptr) noexcept`
- `template<auto __fn, typename _Fn, typename _Up, typename _Td = remove_reference_t<_Up>>`  
`constexpr function_ref (constant_wrapper< __fn, _Fn >, _Up && __ref) noexcept`
- `_Res operator() (_ ArgTypes... __args) const noexcept(_ Noex)`
- `template<typename _Tp>`  
`function_ref & operator= (_Tp)=delete`

## Friends

- `template<typename _Func>`  
`constexpr auto & __polyfunc::__base_of (_Func &) noexcept`
- `template<typename _Func>`  
`constexpr auto & __polyfunc::__invoker_of (_Func &) noexcept`
- `template<typename _Dst, typename _Src>`  
`constexpr bool __polyfunc::__is_funcref_assignable () noexcept`
- `template<typename _Src, typename _Dst>`  
`constexpr bool __polyfunc::__is_invoker_convertible () noexcept`

### 6.460.1 Detailed Description

`template<typename _Res, typename... _ArgTypes, bool _Noex>`  
`class std::function_ref< _Res(_ ArgTypes...) _GLIBCXX_MOF_CV noexcept(_ Noex)>`

Non-owning polymorphic function wrapper.

Since

C++26

The `std::function_ref` class template is a non-owning call wrapper, that refers to a bound object. Using `function_ref` outside of the lifetime of the bound object has undefined behavior. It supports const-qualification and no-throw guarantees. The qualifications and exception-specification of the signature are respected when invoking the reference function.

### 6.460.2 Constructor & Destructor Documentation

#### `function_ref()` [1/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
 _Fn * __fn) [inline], [noexcept]
```

Target and bound object is function pointed by parameter.

#### `function_ref()` [2/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
 _Fn && __f) [inline], [constexpr], [noexcept]
```

Target and bound object is object referenced by parameter.

References `std::addressof()`, and `std::forward()`.

**function\_ref()** [3/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
 constant_wrapper< __fn, _Fn >) [inline], [constexpr], [noexcept]
```

Target object is `__fn`. There is no bound object.

References [std::\\_\\_invoke\(\)](#).

**function\_ref()** [4/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn, typename _Up, typename _Td = remove_reference_t<_Up>>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
 constant_wrapper< __fn, _Fn > ,
 _Up && __ref) [inline], [constexpr], [noexcept]
```

Target object is equivalent to `std::bind_front<_fn>(std::ref(__ref))`. Bound object is object referenced by second parameter.

References [std::addressof\(\)](#).

**function\_ref()** [5/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn, typename _Td>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
 constant_wrapper< __fn, _Fn > ,
 _Td _GLIBCXX_MOF_CV * __ptr) [inline], [constexpr], [noexcept]
```

Target object is equivalent to `std::bind_front<_fn>(__ptr)`. Bound object is object pointed by second parameter (if any).

**6.460.3 Member Function Documentation****operator>()()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
 _ArgTypes... __args) const [inline], [noexcept]
```

Invoke the target object.

The bound object will be invoked using the supplied arguments, and as const or non-const, as dictated by the template arguments of the `function_ref` specialization.

References [std::forward\(\)](#).

The documentation for this class was generated from the following file:

- [funcref\\_impl.h](#)

**6.461 std::future<\_Res> Class Template Reference**

```
#include <future>
```

Inheritance diagram for std::future< \_Res >:



### Public Member Functions

- **future** (const future &)=delete
- **future** (future &&\_\_uf) noexcept
- `_Res` **get** ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&\_\_fut) noexcept
- **shared\_future**< \_Res > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename \_Rep, typename \_Period>  
**future\_status** **wait\_for** (const chrono::duration< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration>  
**future\_status** **wait\_until** (const chrono::time\_point< \_Clock, \_Duration > &\_\_abs) const

### Protected Types

- typedef \_\_future\_base::\_Result< \_Res > & **\_\_result\_type**

### Protected Member Functions

- \_\_result\_type **\_M\_get\_result** () const
- void **\_M\_swap** (\_\_basic\_future &\_\_that) noexcept

### Friends

- template<typename \_Fn, typename... \_Args>  
future< \_\_async\_result\_of< \_Fn, \_Args... > > **async** (**launch**, \_Fn &&, \_Args &&...)
- template<typename>  
class **packaged\_task**
- class **promise**< \_Res >

### 6.461.1 Detailed Description

```
template<typename _Res>
class std::future< _Res >
```

Primary template for future.

### 6.461.2 Constructor & Destructor Documentation

#### future()

```
template<typename _Res>
std::future< _Res >::future (
 future< _Res > && __uf) [inline], [noexcept]
```

Move constructor.

References [std::move\(\)](#).

### 6.461.3 Member Function Documentation

#### M\_get\_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by [std::future< \\_Res >::get\(\)](#), and [std::shared\\_future< \\_Res >::get\(\)](#).

#### get()

```
template<typename _Res>
_Res std::future< _Res >::get () [inline]
```

Retrieving the value.

References [std::\\_\\_basic\\_future< \\_Res >::\\_M\\_get\\_result\(\)](#), and [std::move\(\)](#).

The documentation for this class was generated from the following file:

- [future](#)

## 6.462 std::future< \_Res & > Class Template Reference

```
#include <future>
```

Inheritance diagram for `std::future< _Res & >`:



## Public Member Functions

- **future** (const future &)=delete
- **future** (future &&\_\_uf) noexcept
- **\_Res & get** ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&\_\_fut) noexcept
- **shared\_future**<\_Res & > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- **future\_status wait\_for** (const chrono::duration<\_Rep, \_Period > &\_\_rel) const
- **future\_status wait\_until** (const chrono::time\_point<\_Clock, \_Duration > &\_\_abs) const

## Protected Types

- typedef \_\_future\_base::Result<\_Res > & **\_\_result\_type**

## Protected Member Functions

- **\_\_result\_type \_M\_get\_result** () const
- void **\_M\_swap** (\_\_basic\_future &\_\_that) noexcept

## Friends

- template<typename \_Fn, typename... \_Args>  
future<\_\_async\_result\_of<\_Fn, \_Args... > > **async** (**launch**, \_Fn &&, \_Args &&...)
- template<typename>  
class **packaged\_task**
- class **promise**<\_Res & >
- class **promise**<\_Res >

### 6.462.1 Detailed Description

template<typename \_Res>  
class std::future<\_Res & >

Partial specialization for future<R&>.

### 6.462.2 Constructor & Destructor Documentation

#### future()

```
template<typename _Res>
std::future<_Res & >::future (
 future<_Res & > && __uf) [inline], [noexcept]
```

Move constructor.

References [std::move\(\)](#).

### 6.462.3 Member Function Documentation

#### \_M\_get\_result()

```
__result_type std::__basic_future<_Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by [get\(\)](#).

**get()**

```
template<typename _Res>
_Res & std::future< _Res & >::get () [inline]
```

Retrieving the value.

References [\\_M\\_get\\_result\(\)](#).

The documentation for this class was generated from the following file:

- [future](#)

**6.463 std::future< void > Class Reference**

```
#include <future>
```

Inheritance diagram for `std::future< void >`:

**Public Member Functions**

- **future** (const future &)=delete
- [future](#) (future &&\_\_uf) noexcept
- void [get](#) ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&\_\_fut) noexcept
- [shared\\_future](#)< void > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- [future\\_status](#) **wait\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel) const
- [future\\_status](#) **wait\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs) const

**Protected Types**

- typedef \_\_future\_base::\_Result< void > & **\_\_result\_type**

**Protected Member Functions**

- \_\_result\_type [\\_M\\_get\\_result](#) () const
- void [\\_M\\_swap](#) (\_\_basic\_future &\_\_that) noexcept

## Friends

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename>`  
`class packaged_task`
- `class promise< _Res >`
- `class promise< void >`

### 6.463.1 Detailed Description

Explicit specialization for `future<void>`.

### 6.463.2 Constructor & Destructor Documentation

#### `future()`

```
std::future< void >::future (
 future< void > && __uf) [inline], [noexcept]
```

Move constructor.

References [std::move\(\)](#).

### 6.463.3 Member Function Documentation

#### `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by [get\(\)](#).

#### `get()`

```
void std::future< void >::get () [inline]
```

Retrieving the value.

References [\\_M\\_get\\_result\(\)](#).

The documentation for this class was generated from the following file:

- [future](#)

## 6.464 `std::future_error` Class Reference

```
#include <future>
```

Inheritance diagram for `std::future_error`:



#### Public Member Functions

- **future\_error** ([future\\_errc](#) \_\_errc)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const noexcept

#### Friends

- void **\_\_throw\_future\_error** (int)

#### 6.464.1 Detailed Description

Exception type thrown by futures.

Since

C++11

#### 6.464.2 Member Function Documentation

##### what()

```
virtual const char * std::future_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

The documentation for this class was generated from the following file:

- [future](#)

#### 6.465 std::gamma\_distribution<\_RealType> Class Template Reference

```
#include <random>
```



## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- [gamma\\_distribution](#) ()
- [gamma\\_distribution](#) (\_RealType \_\_alpha\_val, \_RealType \_\_beta\_val=\_RealType(1))
- [gamma\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType [alpha](#) () const
- \_RealType [beta](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator](#)() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::gamma\\_distribution](#)< \_RealType1 > &\_\_x)
- bool [operator==](#) (const [gamma\\_distribution](#) &\_\_d1, const [gamma\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::gamma\\_distribution](#)< \_RealType1 > &\_\_x)

### 6.465.1 Detailed Description

template<typename \_RealType = double>  
class [std::gamma\\_distribution](#)< \_RealType >

A gamma continuous distribution for random numbers.  
The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Since

C++11

### 6.465.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.465.3 Constructor & Destructor Documentation

#### gamma\_distribution() [1/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution () [inline]
```

Constructs a gamma distribution with parameters 1 and 1.

#### gamma\_distribution() [2/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
 _RealType __alpha_val,
 _RealType __beta_val = _RealType(1)) [inline], [explicit]
```

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

### 6.465.4 Member Function Documentation

#### alpha()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::alpha () const [inline]
```

Returns the  $\alpha$  of the distribution.

#### beta()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::beta () const [inline]
```

Returns the  $\beta$  of the distribution.

#### max()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

**operator>() [1/2]**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::gamma_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**operator()() [2/2]**

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

References [std::log\(\)](#), and [std::pow\(\)](#).

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::gamma_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**6.465.5 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::gamma_distribution< _RealType1 > & __x) [friend]
```

Inserts a gamma\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__os</code> | An output stream.                                             |
| <code>__x</code>  | A <code>gamma_distribution</code> random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
 const gamma_distribution< _RealType > & __d1,
 const gamma_distribution< _RealType > & __d2) [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::gamma_distribution< _RealType1 > & __x) [friend]
```

Extracts a `gamma_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                  |
| <code>__x</code>  | A <code>gamma_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.466 `std::geometric_distribution<_IntType>` Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- **geometric\_distribution** (double \_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const geometric\_distribution &\_\_d1, const geometric\_distribution &\_\_d2)

### 6.466.1 Detailed Description

```
template<typename _IntType = int>
class std::geometric_distribution< _IntType >
```

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1 - p)^i$  where  $p$  is the parameter of the distribution.

Since

C++11

### 6.466.2 Member Typedef Documentation

#### result\_type

```
template<typename _IntType = int>
typedef _IntType std::geometric_distribution< _IntType >::result_type
```

The type of the range of the distribution.

### 6.466.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::geometric_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator>\(\)\(\)](#).

#### p()

```
template<typename _IntType = int>
double std::geometric_distribution< _IntType >::p () const [inline]
```

Returns the distribution parameter p.

#### param() [1/2]

```
template<typename _IntType = int>
param_type std::geometric_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

#### param() [2/2]

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the geometric distribution.

#### 6.466.4 Friends And Related Symbol Documentation

##### operator==

```
template<typename _IntType = int>
bool operator== (
 const geometric_distribution< _IntType > & __d1,
 const geometric_distribution< _IntType > & __d2) [friend]
```

Return true if two geometric distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 6.467 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Probe\_Fn **probe\_fn**
- typedef Resize\_Policy **resize\_policy**

#### Public Member Functions

- [gp\\_hash\\_table](#) ()
- [gp\\_hash\\_table](#) (const gp\_hash\_table &other)
- [gp\\_hash\\_table](#) (const hash\_fn &h)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)

- `template<typename It>`  
`gp_hash_table` (It first, It last)
- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h)
- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- `template<typename It>`  
`gp_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- `gp_hash_table` & **operator=** (const `gp_hash_table` &other)
- void **swap** (`gp_hash_table` &other)

### 6.467.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type,
typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

### Template Parameters

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                        |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                        |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                 |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                   |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. |
| <i>Probe_Fn</i>      | Probe functor.                                                                                                                                                                                   |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                    |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true                                          |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                  |

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

### 6.467.2 Constructor & Destructor Documentation

#### `gp_hash_table()` [1/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
```



```
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table () [inline]
```

Default constructor.

#### `gp_hash_table()` [2/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

#### `gp_hash_table()` [3/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

#### `gp_hash_table()` [4/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

#### `gp_hash_table()` [5/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const mapped_fn & mf) [inline]
```

```
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

### gp\_hash\_table() [6/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<←
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb←
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_←
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p,
 const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

### gp\_hash\_table() [7/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<←
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb←
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_←
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

### gp\_hash\_table() [8/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<←
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb←
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_←
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
```

```

 It first,
 It last,
 const hash_fn & h) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

#### `gp_hash_table()` [9/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↵
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↵
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

#### `gp_hash_table()` [10/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↵
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↵
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

#### `gp_hash_table()` [11/12]

```

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↵
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↵
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,

```

```
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

### **gp\_hash\_table() [12/12]**

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash =
detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p,
 const resize_policy & rp) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_<probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## **6.468 \_\_gnu\_pbds::gp\_hash\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



#### 6.468.1 Detailed Description

General-probing hash.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 6.469 `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` Class Template Reference

```
#include <gp_ht_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_`

`_Probe_Fn, Probe_Fn, Resize_Policy >`:



## Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Probe_Fn` **comb\_probe\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `Probe_Fn` **probe\_fn**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **gp\_ht\_map** (const gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)
- **gp\_ht\_map** (const Hash\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &, const Resize\_Policy &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Probe\_Fn & **get\_comb\_probe\_fn** ()
- const Comb\_Probe\_Fn & **get\_comb\_probe\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Probe\_Fn & **get\_probe\_fn** ()
- const Probe\_Fn & **get\_probe\_fn** () const
- Resize\_Policy & **get\_resize\_policy** ()
- const Resize\_Policy & **get\_resize\_policy** () const
- **std::pair**< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class **const\_iterator\_**
- class **iterator\_**

### 6.469.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn,
Probe_Fn, Resize_Policy >
```

A general-probing hash-based container.

#### Template Parameters

|                      |                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                                |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                                |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                               |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                             |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                          |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                    |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> . |
| <i>Probe_Fn</i>      | Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .                                                                                                                                                         |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                   |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

### 6.469.2 Member Enumeration Documentation

#### anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
Value stores hash, true or false.
```

### 6.469.3 Member Function Documentation

#### empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_
Fn, Probe_Fn, Resize_Policy >::empty () const [inline], [nodiscard]
True if size() == 0.
```

#### get\_comb\_probe\_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()
Return current comb_probe_fn.
```



**get\_comb\_probe\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const
Return current const comb_probe_fn.
```

**get\_eq\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()
Return current eq_fn.
```

**get\_eq\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const
Return current const eq_fn.
```

**get\_hash\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()
Return current hash_fn.
```

**get\_hash\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const
Return current const hash_fn.
```

**get\_probe\_fn()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()
Return current probe_fn.
```

**get\_probe\_fn()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const
Return current const probe_fn.
```

**get\_resize\_policy()** [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()
Return current resize_policy.
```

**get\_resize\_policy()** [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const
Return current const resize_policy.
```

The documentation for this class was generated from the following file:

- [gp\\_ht\\_map.hpp](#)

## 6.470 std::chrono::gps\_clock Class Reference

```
#include <chrono>
```

### Public Types

- using **duration**
- using **period**
- using **rep**
- using **time\_point**

### Static Public Member Functions

- template<typename \_Duration>  
static gps\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) > > **from\_utc** (const utc\_time< \_Duration > &\_\_t)
- static time\_point **now** ()
- template<typename \_Duration>  
static utc\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) > > **to\_utc** (const gps\_time< \_Duration > &\_\_t)

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 6.470.1 Detailed Description

A clock that measures GPS time.  
The epoch is 1980-01-06 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

## 6.471 `std::greater<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::greater<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.471.1 Detailed Description

```
template<typename _Tp>
struct std::greater<_Tp>
```

One of the [comparison functors](#).

#### 6.471.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

##### `second_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.472 std::ranges::greater Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using [is\\_transparent](#)

### Static Public Member Functions

- `template<typename _Tp, typename _Up>  
static constexpr bool operator() (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Up >())<  
std::declval< _Tp >()))`

#### 6.472.1 Detailed Description

`ranges::greater` function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 6.473 std::greater< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::greater< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- `template<typename _Tp, typename _Up>  
constexpr auto operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) >  
std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) > std::forward< _Up >(__u))`
- `template<typename _Tp, typename _Up>  
constexpr bool operator() (_Tp *__t, _Up *__u) const noexcept`

### 6.473.1 Detailed Description

One of the [comparison functors](#).

### 6.473.2 Member Typedef Documentation

#### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type  
first\_argument\_type is the type of the first argument

#### result\_type

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type  
result\_type is the return type

#### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.474 std::greater\_equal< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::greater\_equal< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

### 6.474.1 Detailed Description

```
template<typename _Tp>
struct std::greater_equal< _Tp >
```

One of the [comparison functors](#).

### 6.474.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.475 std::ranges::greater\_equal Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using [is\\_transparent](#)

### Static Public Member Functions

- ```
template<typename _Tp, typename _Up>
static constexpr bool operator() (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Tp >())<
std::declval< _Up >()))
```

6.475.1 Detailed Description

`ranges::greater_equal` function object type.

The documentation for this struct was generated from the following file:

- [ranges_cmp.h](#)

6.476 std::greater_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::greater_equal< void >:



Public Types

- typedef void [first_argument_type](#)
- typedef __is_transparent [is_transparent](#)
- typedef bool [result_type](#)
- typedef void [second_argument_type](#)

Public Member Functions

- template<typename _Tp, typename _Up>
constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept([std::forward](#)< _Tp >(__t) >=[std::forward](#)< _Up >(__u))) -> decltype([std::forward](#)< _Tp >(__t) >=[std::forward](#)< _Up >(__u))
- template<typename _Tp, typename _Up>
constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept

6.476.1 Detailed Description

One of the [comparison functors](#).

6.476.2 Member Typedef Documentation

first_argument_type

typedef void [std::binary_function](#)< void, void, bool >::first_argument_type
first_argument_type is the type of the first argument

result_type

typedef bool [std::binary_function](#)< void, void, bool >::result_type
result_type is the return type

second_argument_type

typedef void [std::binary_function](#)< void, void, bool >::second_argument_type
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

6.477 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

`#include <throw_allocator.h>`

Public Member Functions

- `group_adjustor` (size_t size)

6.477.1 Detailed Description

Group condition.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

6.478 `__gnu_parallel::growing_blocks_tag` Struct Reference

`#include <tags.h>`

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



6.478.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

The documentation for this struct was generated from the following file:

- [tags.h](#)

6.479 `std::gslice` Class Reference

`#include <gslice.h>`

Public Member Functions

- [gslice](#) ()
- [gslice](#) (const [gslice](#) &)
- [gslice](#) (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- valarray< size_t > [size](#) () const
- size_t [start](#) () const
- valarray< size_t > [stride](#) () const

Friends

- template<typename _Tp>
class **valarray**

6.479.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size. For example, if you have offset==3, stride[0]==11, size[1]==3, stride[1]==3, then slice[0,0]==array[3], slice[0,1]==array[6], slice[0,2]==array[9], slice[1,0]==array[14], slice[1,1]==array[17], slice[1,2]==array[20].

The documentation for this class was generated from the following file:

- [gslice.h](#)

6.480 std::gslice_array< _Tp > Class Template Reference

```
#include <gslice_array.h>
```

Public Types

- typedef _Tp **value_type**

Public Member Functions

- [gslice_array](#) (const [gslice_array](#) &)
- template<class _Dom>
void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void **operator%=** (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void **operator&=** (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void **operator*=** (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void **operator+=** (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator-=** (const _Expr< _Dom, _Tp > &) const

- void **operator**== (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**/= (const _Expr<_Dom, _Tp> &) const
- void **operator**/= (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**<<= (const _Expr<_Dom, _Tp> &) const
- void **operator**<<= (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**= (const _Expr<_Dom, _Tp> &) const
- void **operator**= (const _Tp &) const
- [gslice_array](#) & **operator**= (const [gslice_array](#) &)
- void **operator**= (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**>>= (const _Expr<_Dom, _Tp> &) const
- void **operator**>>= (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**^= (const _Expr<_Dom, _Tp> &) const
- void **operator**^= (const [valarray](#)<_Tp> &) const
- template<class _Dom>
void **operator**|= (const _Expr<_Dom, _Tp> &) const
- void **operator**|= (const [valarray](#)<_Tp> &) const

Friends

- class [valarray](#)<_Tp>

6.480.1 Detailed Description

template<typename _Tp>
class std::gslice_array<_Tp>

Reference to multi-dimensional subset of an array.

A [gslice_array](#) is a reference to the actual elements of an array specified by a [gslice](#). The way to get a [gslice_array](#) is to call `operator[](gslice)` on a [valarray](#). The returned [gslice_array](#) then permits carrying operations out on the referenced subset of elements in the original [valarray](#). For example, `operator+=(valarray)` will add values to the subset of elements in the underlying [valarray](#) this [gslice_array](#) refers to.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

6.480.2 Member Function Documentation

operator%=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator%= (
    const valarray<_Tp> & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator&= (
    const valarray<_Tp> & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator*= (
    const valarray<_Tp> & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator+= (
    const valarray<_Tp> & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator-= (
    const valarray<_Tp> & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator/= (
    const valarray<_Tp> & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator<<= (
    const valarray<_Tp> & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator>>= (
    const valarray<_Tp> & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<typename _Tp>
void std::gslice_array<_Tp>::operator^= (
    const valarray<_Tp> & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator" |=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator|= (
    const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of v.

The documentation for this class was generated from the following files:

- [valarray](#)
- [gslice_array.h](#)

6.481 std::has_virtual_destructor< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.481.1 Detailed Description

```
template<typename _Tp>
struct std::has_virtual_destructor< _Tp >
```

has_virtual_destructor

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.482 std::hash< _Tp > Struct Template Reference

```
#include <functional_hash.h>
```

6.482.1 Detailed Description

```
template<typename _Tp>
struct std::hash< _Tp >
```

Primary class template hash.

Primary class template hash, usable for enum types only.

The documentation for this struct was generated from the following file:

- [string_view](#)

6.483 std::hash< __debug::bitset< _Nb > > Struct Template Reference

```
#include <bitset>
```

Public Member Functions

- `size_t operator() (const __debug::bitset< _Nb > &__b) const` noexcept

6.483.1 Detailed Description

```
template<size_t _Nb>
struct std::hash< __debug::bitset< _Nb > >
```

std::hash specialization for bitset.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

6.484 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

```
#include <vector>
```

Public Member Functions

- `size_t operator()` (const [__debug::vector< bool, _Alloc >](#) &__b) const noexcept

6.484.1 Detailed Description

`template<typename _Alloc>`

`struct std::hash< __debug::vector< bool, _Alloc > >`

`std::hash` specialization for `vector<bool>`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)

6.485 `std::hash< __gnu_cxx::__u16vstring >` Struct Reference

```
#include <vstring.h>
```

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__u16vstring` &__s) const noexcept

6.485.1 Detailed Description

`std::hash` specialization for `__u16vstring`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

6.486 `std::hash< __gnu_cxx::__u32vstring >` Struct Reference

```
#include <vstring.h>
```

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__u32vstring` &__s) const noexcept

6.486.1 Detailed Description

`std::hash` specialization for `__u32vstring`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

6.487 `std::hash< __gnu_cxx::__vstring >` Struct Reference

```
#include <vstring.h>
```

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__vstring` &__s) const noexcept

6.487.1 Detailed Description

std::hash specialization for __vstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

6.488 std::hash< __gnu_cxx::__wvstring > Struct Reference

```
#include <vstring.h>
```

Public Member Functions

- `size_t operator() (const __gnu_cxx::__wvstring &__s) const` noexcept

6.488.1 Detailed Description

std::hash specialization for __wvstring.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

6.489 std::hash< __gnu_cxx::throw_value_limit > Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for std::hash< __gnu_cxx::throw_value_limit >:



Public Types

- `typedef __gnu_cxx::throw_value_limit argument_type`
- `typedef size_t result_type`

Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_limit &__val) const`

6.489.1 Detailed Description

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

6.489.2 Member Typedef Documentation

`argument_type`

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_t>::argument_type [inherited]
argument_type is the type of the argument
```

`result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

6.490 `std::hash< __gnu_cxx::throw_value_random >` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`__val`) const

6.490.1 Detailed Description

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_random`.

6.490.2 Member Typedef Documentation

argument_type

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random, size_t >::argument_type [inherited]
```

argument_type is the type of the argument

result_type

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t >::result_type [inherited]
```

result_type is the return type

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

6.491 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference

```
#include <string>
```

Inheritance diagram for std::hash< __gnu_debug::basic_string< _CharT > >:



6.491.1 Detailed Description

```
template<typename _CharT>
struct std::hash< __gnu_debug::basic_string< _CharT > >
```

std::hash specialization for __gnu_debug::basic_string.

The documentation for this struct was generated from the following file:

- [debug/string](#)

6.492 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference

```
#include <shared_ptr_base.h>
```

Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const` noexcept

6.492.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp>
struct std::hash<__shared_ptr<_Tp, _Lp>>
```

`std::hash` specialization for `__shared_ptr`.

The documentation for this struct was generated from the following file:

- [shared_ptr_base.h](#)

6.493 `std::hash<_Tp*>` Struct Template Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()(_Tp*__p) const` noexcept

6.493.1 Detailed Description

```
template<typename _Tp>
struct std::hash<_Tp*>
```

Partial specializations for pointer types.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.494 `std::hash<basic_string<char, char_traits<char>, _Alloc>>` Struct Template Reference

```
#include <basic_string.h>
```

Public Member Functions

- `size_t operator()(const _StrT &__s) const` noexcept

6.494.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<basic_string<char, char_traits<char>, _Alloc>>
```

`std::hash` specialization for `string`.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

6.495 `std::hash<basic_string<char16_t, char_traits<char16_t>, _Alloc>>` Struct Template Reference

```
#include <basic_string.h>
```

Public Member Functions

- `size_t operator()(const _StrT &__s) const` noexcept

6.495.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
```

std::hash specialization for u16string.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

6.496 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference

```
#include <basic_string.h>
```

Public Member Functions

- `size_t operator() (const _StrT &__s) const noexcept`

6.496.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >
```

std::hash specialization for u32string.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

6.497 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference

```
#include <basic_string.h>
```

Public Member Functions

- `size_t operator() (const _StrT &__s) const noexcept`

6.497.1 Detailed Description

```
template<typename _Alloc>
struct std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >
```

std::hash specialization for wstring.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

6.498 std::hash< bool > Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (bool __val) const noexcept`

6.498.1 Detailed Description

Explicit specialization for `bool`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.499 `std::hash< char >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (char __val) const noexcept`

6.499.1 Detailed Description

Explicit specialization for `char`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.500 `std::hash< char16_t >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (char16_t __val) const noexcept`

6.500.1 Detailed Description

Explicit specialization for `char16_t`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.501 `std::hash< char32_t >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (char32_t __val) const noexcept`

6.501.1 Detailed Description

Explicit specialization for `char32_t`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.502 `std::hash< double >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (double __val) const noexcept`

6.502.1 Detailed Description

Specialization for double.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.503 `std::hash< error_code >` Struct Reference

```
#include <system_error>
```

Public Member Functions

- `size_t operator() (const error_code &__e) const noexcept`

6.503.1 Detailed Description

`std::hash` specialization for `error_code`.

The documentation for this struct was generated from the following file:

- [system_error](#)

6.504 `std::hash< error_condition >` Struct Reference

```
#include <system_error>
```

Public Member Functions

- `size_t operator() (const error_condition &__e) const noexcept`

6.504.1 Detailed Description

`std::hash` specialization for `error_condition`.

The documentation for this struct was generated from the following file:

- [system_error](#)

6.505 `std::hash< experimental::optional< _Tp > >` Struct Template Reference

```
#include <optional>
```

Public Types

- using `argument_type`
- using `result_type`

Public Member Functions

- `size_t operator() (const experimental::optional< _Tp > &__t) const noexcept(noexcept(hash< _Tp > {}(*__t)))`

6.505.1 Detailed Description

```
template<typename _Tp>
struct std::hash< experimental::optional< _Tp > >
```

std::hash partial specialization for experimental::optional

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

6.506 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

Public Member Functions

- `size_t operator()` (const experimental::shared_ptr< _Tp > &__s) const noexcept

6.506.1 Detailed Description

```
template<typename _Tp>
struct std::hash< experimental::shared_ptr< _Tp > >
```

std::hash specialization for shared_ptr.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared_ptr.h](#)

6.507 std::hash< float > Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()` (float __val) const noexcept

6.507.1 Detailed Description

Specialization for float.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.508 std::hash< int > Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()` (int __val) const noexcept

6.508.1 Detailed Description

Explicit specialization for int.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.509 `std::hash< long >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()` (`long __val`) `const noexcept`

6.509.1 Detailed Description

Explicit specialization for `long`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.510 `std::hash< long double >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()` (`long double __val`) `const noexcept`

6.510.1 Detailed Description

Specialization for `long double`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.511 `std::hash< long long >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator()` (`long long __val`) `const noexcept`

6.511.1 Detailed Description

Explicit specialization for `long long`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.512 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

Public Member Functions

- `size_t operator()` (`const shared_ptr< _Tp > &__s`) `const noexcept`

6.512.1 Detailed Description

```
template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

The documentation for this struct was generated from the following file:

- [bits/shared_ptr.h](#)

6.513 `std::hash< short >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (short __val) const` noexcept

6.513.1 Detailed Description

Explicit specialization for `short`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.514 `std::hash< signed char >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (signed char __val) const` noexcept

6.514.1 Detailed Description

Explicit specialization for `signed char`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.515 `std::hash< thread::id >` Struct Reference

```
#include <std_thread.h>
```

Public Member Functions

- `size_t operator() (const thread::id &__id) const` noexcept

6.515.1 Detailed Description

`std::hash` specialization for `thread::id`.

The documentation for this struct was generated from the following file:

- [std_thread.h](#)

6.516 `std::hash< type_index >` Struct Reference

```
#include <typeindex>
```

Public Types

- typedef [type_index](#) **argument_type**
- typedef size_t **result_type**

Public Member Functions

- size_t **operator()** (const [type_index](#) &__ti) const noexcept

6.516.1 Detailed Description

std::hash specialization for type_index.

The documentation for this struct was generated from the following file:

- [typeindex](#)

6.517 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference

```
#include <unique_ptr.h>
```

6.517.1 Detailed Description

```
template<typename _Tp, typename _Dp>  
struct std::hash< unique_ptr< _Tp, _Dp > >
```

std::hash specialization for unique_ptr.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

6.518 std::hash< unsigned char > Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- size_t **operator()** (unsigned char __val) const noexcept

6.518.1 Detailed Description

Explicit specialization for unsigned char.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.519 std::hash< unsigned int > Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- size_t **operator()** (unsigned int __val) const noexcept

6.519.1 Detailed Description

Explicit specialization for unsigned int.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.520 `std::hash< unsigned long >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (unsigned long __val) const` noexcept

6.520.1 Detailed Description

Explicit specialization for unsigned long.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.521 `std::hash< unsigned long long >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (unsigned long long __val) const` noexcept

6.521.1 Detailed Description

Explicit specialization for unsigned long long.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.522 `std::hash< unsigned short >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (unsigned short __val) const` noexcept

6.522.1 Detailed Description

Explicit specialization for unsigned short.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.523 `std::hash< wchar_t >` Struct Reference

```
#include <functional_hash.h>
```

Public Member Functions

- `size_t operator() (wchar_t __val) const` noexcept

6.523.1 Detailed Description

Explicit specialization for wchar_t.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

6.524 `std::hash<::bitset<_Nb>>` Struct Template Reference

```
#include <bitset>
```

Public Member Functions

- `size_t operator()` (const [::bitset<_Nb>](#) &__b) const noexcept

6.524.1 Detailed Description

```
template<size_t _Nb>  
struct std::hash<::bitset<_Nb>>
```

`std::hash` specialization for `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

6.525 `std::hash<::vector<bool,_Alloc>>` Struct Template Reference

```
#include <stl_bvector.h>
```

Public Member Functions

- `size_t operator()` (const [::vector<bool,_Alloc>](#) &) const noexcept

6.525.1 Detailed Description

```
template<typename _Alloc>  
struct std::hash<::vector<bool,_Alloc>>
```

`std::hash` specialization for `vector<bool>`.

The documentation for this struct was generated from the following files:

- [stl_bvector.h](#)
- [vector.tcc](#)

6.526 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`:



6.526.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

6.527 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

Public Types

- typedef Eq_Fn `eq_fn_base`

- typedef [rebind_traits](#)< _Alloc, Key >::const_reference **key_const_reference**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, key_const_reference r_rhs_key) const
- void **swap** (const hash_eq_fn &other)

6.527.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.
The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

6.528 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef [rebind_traits](#)< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash) const
- void **swap** (const hash_eq_fn &other)

6.528.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.
The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

6.529 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::hash_exponential_size_policy< Size_Type >`:



Public Types

- typedef `Size_Type` **size_type**

Public Member Functions

- [hash_exponential_size_policy](#) (`size_type start_size=8, size_type grow_factor=2`)
- void **swap** ([hash_exponential_size_policy](#) `< Size_Type > &other`)

Protected Member Functions

- `size_type` **get_nearest_larger_size** (`size_type size`) const
- `size_type` **get_nearest_smaller_size** (`size_type size`) const

6.529.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >
```

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

6.529.2 Constructor & Destructor Documentation

hash_exponential_size_policy()

```
template<typename Size_Type>
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
    size_type start_size = 8,
    size_type grow_factor = 2)
```

Default constructor, or onstructor taking a `start_size`, or constructor taking a `start_size` and `grow_factor`. The policy will use the sequence of sizes `start_size`, `start_size* grow_factor`, `start_size* grow_factor^2`, ...

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

6.530 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:



Public Types

- enum { [external_load_access](#) }
- typedef `Size_Type` **size_type**

Public Member Functions

- [hash_load_check_resize_trigger](#) (float load_min=0.125, float load_max=0.5)
- `std::pair< float, float >` [get_loads](#) () const
- void [set_loads](#) (`std::pair< float, float >` load_pair)
- void **swap** ([hash_load_check_resize_trigger](#) &other)

Protected Member Functions

- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void [notify_cleared](#) ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_entries)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void [notify_inserted](#) (size_type num_entries)
- void [notify_resized](#) (size_type new_size)

6.530.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.

6.530.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

Enumerator

<code>external_load_access</code>	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.
-----------------------------------	---

6.530.3 Constructor & Destructor Documentation

`hash_load_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_↵
resize_trigger (
    float load_min = 0.125,
    float load_max = 0.5)
```

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

6.530.4 Member Function Documentation

`get_loads()`

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_↵
_Type >::get_loads () const [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

`notify_cleared()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
cleared () [protected]
```

Notifies the table was cleared.

`notify_inserted()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
inserted (
```

```
    size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is `num_entries`.

`notify_resized()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
resized (
```

```
    size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

References [notify_resized\(\)](#).
 Referenced by [notify_resized\(\)](#).

set_loads()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
    std::pair< float, float > load_pair)
```

Sets the loads through a pair of the minimal and maximal loads, respectively.
 The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

6.531 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >:



6.531.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

The documentation for this class was generated from the following file:

- [hash_load_check_resize_trigger_size_base.hpp](#)

6.532 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference

```
#include <hash_load_check_resize_trigger_size_base.hpp>
```


Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- `size_type` **get_size** () const
- void **set_size** (`size_type` size)
- void **swap** (`hash_load_check_resize_trigger_size_base` &other)

6.532.1 Detailed Description

template<typename `Size_Type`>

class `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base`< `Size_Type`, true >

Specializations.

The documentation for this class was generated from the following file:

- [hash_load_check_resize_trigger_size_base.hpp](#)

6.533 `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

```
#include <hash_map>
```

Public Types

- typedef `_Ht::allocator_type` **allocator_type**
- typedef `_Ht::const_iterator` **const_iterator**
- typedef `_Ht::const_pointer` **const_pointer**
- typedef `_Ht::const_reference` **const_reference**
- typedef `_Tp` **data_type**
- typedef `_Ht::difference_type` **difference_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key_equal**
- typedef `_Ht::key_type` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size_type**
- typedef `_Ht::value_type` **value_type**

Public Member Functions

- template<class `_InputIterator`>
hash_map (`_InputIterator` __f, `_InputIterator` __l)
- template<class `_InputIterator`>
hash_map (`_InputIterator` __f, `_InputIterator` __l, `size_type` __n)
- template<class `_InputIterator`>
hash_map (`_InputIterator` __f, `_InputIterator` __l, `size_type` __n, const hasher &__hf)
- template<class `_InputIterator`>
hash_map (`_InputIterator` __f, `_InputIterator` __l, `size_type` __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())

- **hash_map** (size_type __n)
- **hash_map** (size_type __n, const hasher &__hf)
- **hash_map** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator<__type>())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_map &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _AI>
bool **operator==** (const hash_map< _K1, _T1, _HF, _EqK, _AI > &, const hash_map< _K1, _T1, _HF, _EqK, _AI > &)

6.533.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class
_Alloc = allocator<_Tp>>>
class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

The documentation for this class was generated from the following file:

- [hash_map](#)

6.534 `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

```
#include <hash_map>
```

Public Types

- typedef `_Ht::allocator_type` **allocator_type**
- typedef `_Ht::const_iterator` **const_iterator**
- typedef `_Ht::const_pointer` **const_pointer**
- typedef `_Ht::const_reference` **const_reference**
- typedef `_Tp` **data_type**
- typedef `_Ht::difference_type` **difference_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key_equal**
- typedef `_Ht::key_type` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size_type**
- typedef `_Ht::value_type` **value_type**

Public Member Functions

- template<class `_InputIterator`>
hash_multimap (`_InputIterator __f, _InputIterator __l`)
- template<class `_InputIterator`>
hash_multimap (`_InputIterator __f, _InputIterator __l, size_type __n`)
- template<class `_InputIterator`>
hash_multimap (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- template<class `_InputIterator`>
hash_multimap (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash_multimap** (`size_type __n`)
- **hash_multimap** (`size_type __n, const hasher &__hf`)
- **hash_multimap** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)

- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multimap &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al>
bool **operator==** (const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &, const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &)

6.534.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class **__gnu_cxx::hash_multimap**< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

The documentation for this class was generated from the following file:

- [hash_map](#)

6.535 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

```
#include <hash_set>
```

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- `template<class _InputIterator>`
hash_multiset (`_InputIterator __f, _InputIterator __l`)
- `template<class _InputIterator>`
hash_multiset (`_InputIterator __f, _InputIterator __l, size_type __n`)
- `template<class _InputIterator>`
hash_multiset (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- `template<class _InputIterator>`
hash_multiset (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash_multiset** (`size_type __n`)
- **hash_multiset** (`size_type __n, const hasher &__hf`)
- **hash_multiset** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** () const
- `size_type` **bucket_count** () const
- void **clear** ()
- `size_type` **count** (`const key_type &__key`) const
- `size_type` **elems_in_bucket** (`size_type __n`) const
- bool **empty** () const
- iterator **end** () const
- `pair< iterator, iterator >` **equal_range** (`const key_type &__key`) const
- `size_type` **erase** (`const key_type &__key`)
- void **erase** (`iterator __f, iterator __l`)
- void **erase** (`iterator __it`)
- iterator **find** (`const key_type &__key`) const
- `allocator_type` **get_allocator** () const
- hasher **hash_funct** () const
- `template<class _InputIterator>`
void **insert** (`_InputIterator __f, _InputIterator __l`)
- iterator **insert** (`const value_type &__obj`)
- iterator **insert_noresize** (`const value_type &__obj`)
- `key_equal` **key_eq** () const
- `size_type` **max_bucket_count** () const
- `size_type` **max_size** () const
- void **resize** (`size_type __hint`)
- `size_type` **size** () const
- void **swap** (`hash_multiset &hs`)

Friends

- `template<class _Val, class _HF, class _EqK, class _Al>`
bool **operator==** (`const hash_multiset<_Val, _HF, _EqK, _Al> &, const hash_multiset<_Val, _HF, _EqK, _Al> &`)

6.535.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc
= allocator<_Value>>
class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

The documentation for this class was generated from the following file:

- [hash_set](#)

6.536 __gnu_pbds::hash_prime_size_policy Class Reference

```
#include <hash_policy.hpp>
```

Public Types

- typedef std::size_t [size_type](#)

Public Member Functions

- [hash_prime_size_policy](#) ([size_type](#) start_size=8)
- void **swap** ([hash_prime_size_policy](#) &other)

Protected Member Functions

- [size_type](#) **get_nearest_larger_size** ([size_type](#) size) const
- [size_type](#) **get_nearest_smaller_size** ([size_type](#) size) const

6.536.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

6.536.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

6.536.3 Constructor & Destructor Documentation

hash_prime_size_policy()

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (  
    size\_type start_size = 8) [inline]
```

Default constructor, or onstructor taking a start_size The policy will use the sequence of sizes approximately start_size, start_size* 2, start_size* 2^2, ...

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

6.537 __gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

```
#include <hash_set>
```

Public Types

- typedef `_Ht::allocator_type` **allocator_type**
- typedef `_Ht::const_iterator` **const_iterator**
- typedef `_Alloc_traits::const_pointer` **const_pointer**
- typedef `_Alloc_traits::const_reference` **const_reference**
- typedef `_Ht::difference_type` **difference_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::const_iterator` **iterator**
- typedef `_Ht::key_equal` **key_equal**
- typedef `_Ht::key_type` **key_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Ht::size_type` **size_type**
- typedef `_Ht::value_type` **value_type**

Public Member Functions

- template<class `_InputIterator`>
hash_set (`_InputIterator __f, _InputIterator __l`)
- template<class `_InputIterator`>
hash_set (`_InputIterator __f, _InputIterator __l, size_type __n`)
- template<class `_InputIterator`>
hash_set (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- template<class `_InputIterator`>
hash_set (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- **hash_set** (`size_type __n`)
- **hash_set** (`size_type __n, const hasher &__hf`)
- **hash_set** (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator **begin** () const
- `size_type` **bucket_count** () const
- void **clear** ()
- `size_type` **count** (`const key_type &__key`) const
- `size_type` **elems_in_bucket** (`size_type __n`) const
- bool **empty** () const
- iterator **end** () const
- `pair< iterator, iterator >` **equal_range** (`const key_type &__key`) const
- `size_type` **erase** (`const key_type &__key`)
- void **erase** (`iterator __f, iterator __l`)
- void **erase** (`iterator __it`)
- iterator **find** (`const key_type &__key`) const
- `allocator_type` **get_allocator** () const
- hasher **hash_funct** () const
- template<class `_InputIterator`>
void **insert** (`_InputIterator __f, _InputIterator __l`)
- `pair< iterator, bool >` **insert** (`const value_type &__obj`)
- `pair< iterator, bool >` **insert_noresize** (`const value_type &__obj`)
- `key_equal` **key_eq** () const
- `size_type` **max_bucket_count** () const
- `size_type` **max_size** () const
- void **resize** (`size_type __hint`)
- `size_type` **size** () const
- void **swap** (`hash_set &__hs`)

Friends

- `template<class _Val, class _HF, class _EqK, class _Al>`
`bool operator== (const hash_set< _Val, _HF, _EqK, _Al > &, const hash_set< _Val, _HF, _EqK, _Al > &)`

6.537.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc
= allocator<_Value>>
class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html

The documentation for this class was generated from the following file:

- [hash_set](#)

6.538 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >`:



Public Types

- enum { [external_load_access](#) }
- enum { [external_size_access](#) }
- typedef Size_Policy [size_policy](#)
- typedef Size_Type [size_type](#)
- typedef Trigger_Policy [trigger_policy](#)

Public Member Functions

- [hash_standard_resize_policy](#) ()
- [hash_standard_resize_policy](#) (const Size_Policy &r_size_policy)
- [hash_standard_resize_policy](#) (const Size_Policy &r_size_policy, const Trigger_Policy &r_trigger_policy)
- size_type [get_actual_size](#) () const
- `std::pair< float, float >` [get_loads](#) () const
- Size_Policy & [get_size_policy](#) ()

- `const Size_Policy & get_size_policy () const`
- `Trigger_Policy & get_trigger_policy ()`
- `const Trigger_Policy & get_trigger_policy () const`
- `void resize (size_type suggested_new_size)`
- `void set_loads (std::pair< float, float > load_pair)`
- `void swap (hash_exponential_size_policy< Size_Type > &other)`
- `void swap (hash_load_check_resize_trigger &other)`
- `void swap (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)`

Protected Member Functions

- `size_type get_nearest_larger_size (size_type size) const`
- `size_type get_nearest_smaller_size (size_type size) const`
- `size_type get_new_size (size_type size, size_type num_used_e) const`
- `bool is_grow_needed (size_type size, size_type num_entries) const`
- `bool is_resize_needed () const`
- `void notify_cleared ()`
- `void notify_erase_search_collision ()`
- `void notify_erase_search_end ()`
- `void notify_erase_search_start ()`
- `void notify_erased (size_type num_e)`
- `void notify_externally_resized (size_type new_size)`
- `void notify_find_search_collision ()`
- `void notify_find_search_end ()`
- `void notify_find_search_start ()`
- `void notify_insert_search_collision ()`
- `void notify_insert_search_end ()`
- `void notify_insert_search_start ()`
- `void notify_inserted (size_type num_e)`
- `void notify_resized (size_type new_size)`

6.538.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_↵
check_resize_trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_↵
Type >
```

A resize policy which delegates operations to size and trigger policies.

6.538.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum [inherited]
```

Enumerator

<code>external_load_access</code>	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.
-----------------------------------	---

6.538.3 Constructor & Destructor Documentation

hash_standard_resize_policy() [1/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy ()
```

Default constructor.

hash_standard_resize_policy() [2/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy)
```

constructor taking some policies r_size_policy will be copied by the Size_Policy object of this object.

hash_standard_resize_policy() [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↵
_Type >::hash_standard_resize_policy (
    const Size_Policy & r_size_policy,
    const Trigger_Policy & r_trigger_policy)
```

constructor taking some policies. r_size_policy will be copied by the Size_Policy object of this object. r_trigger_policy will be copied by the Trigger_Policy object of this object.

6.538.4 Member Function Documentation

get_actual_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::↵
size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size↵
Access, Size_Type >::get_actual_size () const [inline]
```

Returns the actual size of the container.

get_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↵
_Type >::get_loads () const [inline], [inherited]
```

Returns a pair of the minimal and maximal loads, respectively.

get_new_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↵
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::↵
size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size↵
Access, Size_Type >::get_new_size (
```

```
size_type size,
size_type num_used_e) const [protected]
```

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

`get_size_policy()` [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_size_policy ()
```

Access to the `Size_Policy` object used.

`get_size_policy()` [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
_Size_Access, Size_Type >::get_size_policy () const
```

Const access to the `Size_Policy` object used.

`get_trigger_policy()` [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_trigger_policy ()
```

Access to the `Trigger_Policy` object used.

`get_trigger_policy()` [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy () const
```

Access to the `Trigger_Policy` object used.

`resize()`

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type>
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
```

```
size_type suggested_new_size)
```

Resizes the container to `suggested_new_size`, a suggested size (the actual size will be determined by the `Size_Policy` object).

`set_loads()`

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
std::pair< float, float > load_pair) [inherited]
```

Sets the loads through a pair of the minimal and maximal loads, respectively.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

6.539 `std::chrono::hh_mm_ss<_Duration>` Class Template Reference

```
#include <chrono>
```

Public Types

- using **precision**

Public Member Functions

- constexpr **hh_mm_ss** (`_Duration __d`)
- constexpr `chrono::hours` **hours** () const noexcept
- constexpr bool **is_negative** () const noexcept
- constexpr `chrono::minutes` **minutes** () const noexcept
- constexpr **operator precision** () const noexcept
- constexpr `chrono::seconds` **seconds** () const noexcept
- constexpr precision **subseconds** () const noexcept
- constexpr precision **to_duration** () const noexcept

Static Public Attributes

- static constexpr unsigned **fractional_width**

6.539.1 Detailed Description

```
template<typename _Duration>  
class std::chrono::hh_mm_ss<_Duration>
```

Utility for splitting a duration into hours, minutes, and seconds

This is a convenience type that provides accessors for the constituent parts (hours, minutes, seconds and subseconds) of a duration.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

6.540 `std::locale::id` Class Reference

```
#include <locale_classes.h>
```

Public Member Functions

- `id` ()
- `size_t _M_id` () const throw ()

Friends

- template<typename _Facet>
const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Facet>
bool **has_facet** (const locale &__loc) throw ()
- class **locale**
- class **locale::_Impl**
- template<typename _Facet>
const _Facet & **use_facet** (const locale &__loc)

6.540.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

6.540.2 Constructor & Destructor Documentation

id()

```
std::locale::id::id () [inline]
```

Constructor.

6.540.3 Friends And Related Symbol Documentation

has_facet

```
template<typename _Facet>
bool has_facet (
    const locale & __loc) throw ( ) [friend]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

Returns

true if `__loc` contains a facet of type `_Facet`, else false.

use_facet

```
template<typename _Facet>
const _Facet & use_facet (
    const locale & __loc) [friend]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

Returns

Reference to facet of type `Facet`.

Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
--	---

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

6.541 std::thread::id Class Reference

```
#include <thread>
```

Public Member Functions

- `id` (`native_handle_type __id`)

Friends

- struct `hash< id >`
- template<class `_CharT`, class `_Traits`>
[`basic_ostream`](#)< `_CharT`, `_Traits` > & `operator`<< ([`basic_ostream`](#)< `_CharT`, `_Traits` > &__out, `id __id`)
- strong_ordering `operator`<=> (`id __x`, `id __y`) noexcept
- bool `operator`== (`id __x`, `id __y`) noexcept
- class `thread`

Related Symbols

(Note that these are not member symbols.)

- bool `operator`== ([`thread::id`](#) __x, [`thread::id`](#) __y) noexcept
- strong_ordering `operator`<=> ([`thread::id`](#) __x, [`thread::id`](#) __y) noexcept
- template<class `_CharT`, class `_Traits`>
[`basic_ostream`](#)< `_CharT`, `_Traits` > & `operator`<< ([`basic_ostream`](#)< `_CharT`, `_Traits` > &__out, [`thread::id`](#) __id)

6.541.1 Detailed Description

A `std::thread::id` is a unique identifier for a thread.

Since

C++11

The documentation for this class was generated from the following files:

- [std_thread.h](#)
- [thread](#)

6.542 std::identity Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using `is_transparent`

Static Public Member Functions

- `template<typename _Tp>`
`static constexpr _Tp && operator() (_Tp &&__t) noexcept`

6.542.1 Detailed Description

[func.identity] The identity function.

The documentation for this struct was generated from the following file:

- [ranges_cmp.h](#)

6.543 std::experimental::fundamentals_v1::in_place_t Struct Reference

```
#include <optional>
```

6.543.1 Detailed Description

Tag type for in-place construction.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

6.544 std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType > Class Template Reference

```
#include <random>
```

Public Types

- typedef _UIntType [result_type](#)

Public Member Functions

- [independent_bits_engine](#) ()
- [independent_bits_engine](#) (_RandomNumberEngine &&__rng)
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`
[independent_bits_engine](#) (_Sseq &__q)
- [independent_bits_engine](#) (const _RandomNumberEngine &__rng)
- [independent_bits_engine](#) ([result_type](#) __s)
- const _RandomNumberEngine & [base](#) () const noexcept
- void [discard](#) (unsigned long long __z)
- [result_type](#) [operator\(\)](#) ()
- void [seed](#) ()
- `template<typename _Sseq>`
`_If_seed_seq<_Sseq> > seed (_Sseq &__q)`
- void [seed](#) ([result_type](#) __s)

Static Public Member Functions

- static constexpr [result_type](#) max ()
- static constexpr [result_type](#) min ()

Friends

- bool [operator==](#) (const [independent_bits_engine](#) &__lhs, const [independent_bits_engine](#) &__rhs)
- template<typename _CharT, typename _Traits>
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &__is,
[std::independent_bits_engine](#)< _RandomNumberEngine, __w, _UIntType > &__x)

6.544.1 Detailed Description

template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class [std::independent_bits_engine](#)< _RandomNumberEngine, __w, _UIntType >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specified number of bits __w.

Since

C++11

6.544.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
typedef _UIntType std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::result_type
```

The type of the generated random value.

6.544.3 Constructor & Destructor Documentation

[independent_bits_engine](#)() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ()
[inline]
```

Constructs a default [independent_bits_engine](#) engine.

The underlying engine is default constructed as well.

Referenced by [operator==](#).

[independent_bits_engine](#)() [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a [independent_bits_engine](#) engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

independent_bits_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

References [std::move\(\)](#).

independent_bits_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    result_type __s) [inline], [explicit]
```

Seed constructs a `independent_bits_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

independent_bits_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
    _Sseq & __q) [inline], [explicit]
```

Generator construct a `independent_bits_engine` engine.

Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

6.544.4 Member Function Documentation**base()**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine & std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::base () const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
```

```
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (
    unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::max ()
[inline], [static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::min ()
[inline], [static], [constexpr]
```

Gets the minimum value in the generated random number range.

operator>()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()
```

Gets the next value in the generated random number sequence.

References [std::__lg\(\)](#), [std::__numeric_limits_base::digits](#), and [std::numeric_limits< _Tp >::max\(\)](#).

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed () [inline]
```

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed
(
    _Sseq & __q) [inline]
```

Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

<code>__q</code>	A seed generator function.
----------------------------------	----------------------------

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
    result_type __s) [inline]
```

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

6.544.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
    const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
    const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs) [friend]
```

Compares two independent_bits_engine random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

References [independent_bits_engine\(\)](#).

operator>>

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x) [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

6.545 std::indirect_array<_Tp> Class Template Reference

```
#include <indirect_array.h>
```

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- [indirect_array](#) (const indirect_array &)
- template<class _Dom>
void **operator%_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator%_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator&_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator&_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator*_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator*_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator+_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator+_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator-_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator-_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator/_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator/_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator<<_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator<<_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator=** (const _Expr< _Dom, _Tp > &) const
- void **operator=** (const _Tp &) const
- [indirect_array](#) & **operator=** (const [indirect_array](#) &)
- void **operator=** (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator>>_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator>>_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator^_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator^_{2D}**= (const [valarray](#)< _Tp > &) const
- template<class _Dom>
void **operator|_{2D}**= (const _Expr< _Dom, _Tp > &) const
- void **operator|_{2D}**= (const [valarray](#)< _Tp > &) const

Friends

- class [gslice_array](#)< _Tp >
- class [valarray](#)< _Tp >

6.545.1 Detailed Description

template<class _Tp>
class std::indirect_array< _Tp >

Reference to arbitrary subset of an array.

An indirect_array is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an indirect_array is to call operator[](valarray<size_t>) on a valarray. The returned indirect_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

6.545.2 Member Function Documentation

`operator%=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator%= (
    const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

`operator&=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator&= (
    const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

`operator*=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator*= (
    const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

`operator+=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator+= (
    const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

`operator-=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator-= (
    const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

`operator/=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator/= (
    const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

`operator<=<=()`

```
template<class _Tp>
void std::indirect_array< _Tp >::operator<=<= (
```

```
const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<class _Tp>
```

```
void std::indirect_array< _Tp >::operator>>= (
```

```
const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<class _Tp>
```

```
void std::indirect_array< _Tp >::operator^= (
```

```
const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator" |=()

```
template<class _Tp>
```

```
void std::indirect_array< _Tp >::operator|= (
```

```
const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [indirect_array.h](#)

6.546 std::initializer_list< _E > Class Template Reference

```
#include <initializer_list>
```

Public Types

- typedef const _E * **const_iterator**
- typedef const _E & **const_reference**
- typedef const _E * **iterator**
- typedef const _E & **reference**
- typedef size_t **size_type**
- typedef _E **value_type**

Public Member Functions

- constexpr const_iterator **begin** () const noexcept
- constexpr const value_type * **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **size** () const noexcept

6.546.1 Detailed Description

```
template<class _E>
```

```
class std::initializer_list< _E >
```

```
initializer_list
```

The documentation for this class was generated from the following file:

- [initializer_list](#)

6.547 std::inout_ptr_t< _Smart, _Pointer, _Args > Class Template Reference

```
#include <>>
```

Public Member Functions

- **inout_ptr_t** (_Smart &__smart, _Args... __args)
- **inout_ptr_t** (const inout_ptr_t &)=delete
- **operator _Pointer** * () const noexcept
- **operator void **** () const noexcept

6.547.1 Detailed Description

```
template<typename _Smart, typename _Pointer, typename... _Args>  
class std::inout_ptr_t< _Smart, _Pointer, _Args >
```

Smart pointer adaptor for functions taking an inout pointer parameter.

Template Parameters

<i>_Smart</i>	The type of pointer to adapt.
<i>_Pointer</i>	The type of pointer to convert to.
<i>_Args...</i>	Argument types used when resetting the smart pointer.

Since

C++23

The documentation for this class was generated from the following file:

- [out_ptr.h](#)

6.548 std::input_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::input_iterator_tag`:



6.548.1 Detailed Description

Marking input iterators.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

6.549 __gnu_pbds::insert_error Struct Reference

```
#include <exception.hpp>
```


Inheritance diagram for `__gnu_pbds::insert_error`:



Public Member Functions

- virtual const char * [what](#) () const noexcept

6.549.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

6.549.2 Member Function Documentation

`what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

6.550 `std::insert_iterator<_Container>` Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::insert_iterator< _Container >`:



Public Types

- typedef `_Container` `container_type`
- using `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- constexpr `insert_iterator` (`_Container &__x`, `_Iter __i`)
- constexpr `insert_iterator` & `operator*` ()
- constexpr `insert_iterator` & `operator++` ()
- constexpr `insert_iterator` & `operator++` (int)
- constexpr `insert_iterator` & `operator=` (const typename `_Container::value_type` &__value)
- constexpr `insert_iterator` & `operator=` (typename `_Container::value_type` &&__value)

Protected Attributes

- `_Container *` `container`
- `_Iter` `iter`

6.550.1 Detailed Description

```
template<typename _Container>
class std::insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

6.550.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
iterator_category [inherited]
```

One of the [tag types](#).

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

6.550.3 Constructor & Destructor Documentation

insert_iterator()

```
template<typename _Container>
std::insert_iterator< _Container >::insert_iterator (
    _Container & __x,
    _Iter __i) [inline], [constexpr]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

References [std::__addressof\(\)](#).

6.550.4 Member Function Documentation

operator*()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator* () [inline], [nodiscard], [constexpr]
```

Simply returns *this.

operator++() [1/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ () [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*).

operator++() [2/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ (
    int ) [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*).

operator=()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator= (
    const typename _Container::value_type & __value) [inline], [constexpr]
```

Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container<T></code> .
----------------------	---

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;
```

```
// vector v contains A, 1, 2, 3, and Z
```

The documentation for this class was generated from the following file:

- [bits/stl_iterator.h](#)

6.551 std::integer_sequence< _Tp, _Idx > Struct Template Reference

```
#include <utility.h>
```

Public Types

- `typedef _Tp value_type`

Static Public Member Functions

- `static constexpr size_t size () noexcept`

6.551.1 Detailed Description

```
template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence< _Tp, _Idx >
```

Class template `integer_sequence`.

The documentation for this struct was generated from the following file:

- [utility.h](#)

6.552 std::integral_constant< _Tp, __v > Struct Template Reference

```
#include <type_traits>
```

Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr _Tp **value**

6.552.1 Detailed Description

```
template<typename _Tp, _Tp __v>  
struct std::integral_constant< _Tp, __v >
```

integral_constant

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.553 std::invalid_argument Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const char *)
- **invalid_argument** (const invalid_argument &)=default

- **invalid_argument** (const [string](#) &__arg)
- **invalid_argument** (invalid_argument &&)=default
- invalid_argument & **operator=** (const invalid_argument &)=default
- invalid_argument & **operator=** (invalid_argument &&)=default
- virtual const char * [what](#) () const noexcept

6.553.1 Detailed Description

Thrown to report invalid arguments to functions.

6.553.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept_except.h](#)

6.554 std::ios_base Class Reference

```
#include <ios_base.h>
```

Inheritance diagram for std::ios_base:



Classes

- class [failure](#)

Public Types

- enum [event](#) { [erase_event](#) , [imbue_event](#) , [copyfmt_event](#) }
- typedef void(* [event_callback](#)) ([event](#) __e, [ios_base](#) &__b, int __i)
- typedef [_ios_Fmtflags](#) [fmtflags](#)

- typedef `_ios_istate` [istate](#)
- typedef `_ios_Openmode` [openmode](#)
- typedef `_ios_Seekdir` [seekdir](#)

Public Member Functions

- **ios_base** (const ios_base &)=delete
- virtual [~ios_base](#) ()
- const [locale](#) & [_M_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc) throw ()
- long & [iword](#) (int __ix)
- ios_base & **operator=** (const ios_base &)=delete
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- void *& [pword](#) (int __ix)
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- void [unsetf](#) ([fmtflags](#) __mask)
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) __wide)

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [openmode](#) [__noreplace](#)
- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [istate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [istate](#) [eofbit](#)
- static const [istate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [istate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)

- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_call_callbacks](#) ([event __ev](#)) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words & _M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- void [_M_move](#) (ios_base &) noexcept
- void [_M_swap](#) (ios_base &__rhs) noexcept

Protected Attributes

- [_Callback_list * _M_callbacks](#)
- [iostate _M_exception](#)
- [fmtflags _M_flags](#)
- [locale _M_ios_locale](#)
- [_Words _M_local_word](#) [[_S_local_word_size](#)]
- [streamsize _M_precision](#)
- [iostate _M_streambuf_state](#)
- [streamsize _M_width](#)
- [_Words * _M_word](#)
- int [_M_word_size](#)
- [_Words _M_word_zero](#)

6.554.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

6.554.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

Parameters

 <code>__e</code>	One of the members of the event enum.
 <code>__b</code>	Reference to the ios_base object.
 <code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`

- basefield
- floatfield

iostate

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

6.554.3 Member Enumeration Documentation

event

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

6.554.4 Constructor & Destructor Documentation

`~ios_base()`

```
virtual std::ios_base::~~ios_base () [virtual]
```

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

6.554.5 Member Function Documentation

`_M_getloc()`

```
const locale & std::ios_base::_M_getloc () const [inline]
```

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Referenced by `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_get<_CharT, _Inlter>::do_get_year()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::time_put<_CharT, _Outlter>::do_put()`, `std::time_get<_CharT, _Inlter>::get()`, and `std::time_put<_CharT, _Outlter>::put()`.

`flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [nodiscard]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_put()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::chrono::operator<<()`, `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator<<()`, `std::__detail::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::operator>>()`, and `std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>::operator>>()`.

`flags()` [2/2]

```
fmtflags std::ios_base::flags (
    fmtflags __fmtfl) [inline]
```

Setting new format flags all at once.

Parameters

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

getloc()

```
locale std::ios_base::getloc () const [inline], [nodiscard]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by [std::basic_ios<_CharT, _Traits>::copyfmt\(\)](#), [std::money_put<_CharT, _OutIter>::do_put\(\)](#), [std::basic_ios<_CharT, _Traits>::operator<<\(\)](#), [std::operator>>\(\)](#), [std::operator>>\(\)](#), and [std::ws\(\)](#).

imbue()

```
locale std::ios_base::imbue (
    const locale & __loc) throw ( )
```

Setting a new locale.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Referenced by [std::basic_ios<_CharT, _Traits>::imbue\(\)](#).

isword()

```
long & std::ios_base::isword (
    int __ix) [inline]
```

Access to integer array.

Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

Returns

A reference to an integer associated with the index.

The `isword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

References [isword\(\)](#).

Referenced by [isword\(\)](#).

precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [nodiscard]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Referenced by [std::basic_ios<_CharT, _Traits>::copyfmt\(\)](#), and [std::chrono::operator<<\(\)](#).

precision() [2/2]

```
streamsize std::ios_base::precision (  
    streamsize __prec) [inline]
```

Changing flags.

Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

Returns

The previous value of `precision()`.

pword()

```
void *& std::ios_base::pword (  
    int __ix) [inline]
```

Access to void pointer array.

Parameters

<code>↵ __ix</code>	Index into the array.
-------------------------	-----------------------

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

register_callback()

```
void std::ios_base::register_callback (  
    event_callback __fn,  
    int __index)
```

Add the callback `__fn` with parameter `__index`.

Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

setf() [1/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl) [inline]
```

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set. Referenced by [std::__detail::operator>>\(\)](#).

setf() [2/2]

```
fmtflags std::ios_base::setf (
    fmtflags __fmtfl,
    fmtflags __mask) [inline]
```

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <i>fmtfl</i> .

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
    bool __sync = true) [static]
```

Interaction with the standard C I/O objects.

Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

unsetf()

```
void std::ios_base::unsetf (
    fmtflags __mask) [inline]
```

Clearing format flags.

Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

width() [1/2]

```
streamsize std::ios_base::width () const [inline], [nodiscard]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, `std::operator>>()`, and `std::operator>>()`.

width() [2/2]

```
streamsize std::ios_base::width (
    streamsize __wide) [inline]
```

Changing flags.

Parameters

<code>__wide</code>	The new width value.
---------------------	----------------------

Returns

The previous value of `width()`.

xalloc()

```
int std::ios_base::xalloc () throw ( ) [static]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

6.554.6 Member Data Documentation**adjustfield**

```
const fmtflags std::ios_base::adjustfield [static]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

app

```
const openmode std::ios_base::app [static]
```

Seek to end before each write.

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_filebuf<_CharT, _Traits>::overflow()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

ate

```
const openmode std::ios_base::ate [static]
```

Open and seek to end immediately after opening.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

badbit

```
const iostate std::ios_base::badbit [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Referenced by `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_ostream<_CharT, _Traits>::sentry::~sentry()`, `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_iostream<char_type, traits_type>::operator<<()`, `std::basic_iostream<char_type, traits_type>::operator>>()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_iostream<char_type, traits_type>::seekg()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_istream<_CharT, _Traits>::unset()`, and `std::ws()`.

basefield

```
const fmtflags std::ios_base::basefield [static]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::oct()`.

beg

```
const seekdir std::ios_base::beg [static]
```

Request a seek relative to the beginning of the stream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::seekpos()`.

binary

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

boolalpha

```
const fmtflags std::ios_base::boolalpha [static]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

cur

```
const seekdir std::ios_base::cur [static]
```

Request a seek relative to the current position within the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_filebuf< _CharT, _Traits >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

dec

```
const fmtflags std::ios_base::dec [static]
```

Converts integer input or generates integer output in decimal base.

Referenced by `std::dec()`, `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<()`, and `std::phiox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>()`.

end

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

eofbit

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_iostream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_iostream< char_type, traits_type >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

failbit

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by [std::basic_ostream<_CharT, _Traits>::sentry::sentry\(\)](#), [std::basic_fstream<char>::close\(\)](#), [std::basic_ifstream<char>::close\(\)](#), [std::basic_ofstream<char>::close\(\)](#), [std::num_get<_CharT, _InIter>::do_get\(\)](#), [std::time_get<_CharT, _InIter>::do_get_monthname\(\)](#), [std::time_get<_CharT, _InIter>::do_get_weekday\(\)](#), [std::time_get<_CharT, _InIter>::do_get_year\(\)](#), [std::basic_iostream<char_type, traits_type>::get\(\)](#), [std::basic_istream<_CharT, _Traits>::get\(\)](#), [std::time_get<_CharT, _InIter>::get\(\)](#), [std::basic_fstream<char>::open\(\)](#), [std::basic_fstream<char>::open\(\)](#), [std::basic_ifstream<char>::open\(\)](#), [std::basic_ifstream<char>::open\(\)](#), [std::basic_ofstream<char>::open\(\)](#), [std::basic_ofstream<char>::open\(\)](#), [std::operator<<\(\)](#), [std::basic_iostream<char_type, traits_type>::operator>>\(\)](#), [std::basic_istream<_CharT, _Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic_iostream<char_type, traits_type>::seekg\(\)](#), [std::basic_istream<_CharT, _Traits>::seekg\(\)](#), [std::basic_istream<_CharT, _Traits>::seekg\(\)](#), [std::basic_ostream<_CharT, _Traits>::seekp\(\)](#), and [std::basic_ostream<_CharT, _Traits>::seekp\(\)](#).

fixed

```
const fmtflags std::ios_base::fixed [static]
```

Generate floating-point output in fixed-point notation.

Referenced by [std::fixed\(\)](#), and [std::hexfloat\(\)](#).

floatfield

```
const fmtflags std::ios_base::floatfield [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by [std::defaultfloat\(\)](#), [std::fixed\(\)](#), [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

goodbit

```
const iostate std::ios_base::goodbit [static]
```

Indicates all is well.

Referenced by [std::basic_istream<_CharT, _Traits>::sentry::sentry\(\)](#), [std::num_get<_CharT, _InIter>::do_get\(\)](#), [std::time_get<_CharT, _InIter>::do_get\(\)](#), [std::time_get<_CharT, _InIter>::do_get_monthname\(\)](#), [std::time_get<_CharT, _InIter>::do_get_year\(\)](#), [std::basic_ostream<_CharT, _Traits>::flush\(\)](#), [std::basic_istream<_CharT, _Traits>::get\(\)](#), [std::basic_istream<_CharT, _Traits>::get\(\)](#), [std::basic_istream<_CharT, _Traits>::get\(\)](#), [std::time_get<_CharT, _InIter>::get\(\)](#), [std::basic_istream<_CharT, _Traits>::getline\(\)](#), [std::basic_istream<_CharT, _Traits>::ignore\(\)](#), [std::basic_istream<_CharT, _Traits>::ignore\(\)](#), [std::basic_istream<_CharT, _Traits>::ignore\(\)](#), [std::basic_iostream<char_type, traits_type>::operator<<\(\)](#), [std::basic_iostream<char_type, traits_type>::operator>>\(\)](#), [std::basic_istream<_CharT, _Traits>::operator>>\(\)](#), [std::basic_istream<_CharT, _Traits>::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic_ostream<_CharT, _Traits>::put\(\)](#), [std::basic_istream<_CharT, _Traits>::putback\(\)](#), [std::basic_istream<_CharT, _Traits>::seekg\(\)](#), [std::basic_istream<_CharT, _Traits>::seekg\(\)](#), [std::basic_istream<_CharT, _Traits>::seekg\(\)](#), [std::basic_istream<_CharT, _Traits>::sync\(\)](#), [std::basic_istream<_CharT, char_traits<_CharT>>::tellg\(\)](#), [std::basic_istream<_CharT, _Traits>::ws\(\)](#), and [std::ws\(\)](#).

hex

```
const fmtflags std::ios_base::hex [static]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by [std::num_get<_CharT, _InIter>::do_get\(\)](#), [std::num_put<_CharT, _OutIter>::do_put\(\)](#), and [std::hex\(\)](#).

in

```
const openmode std::ios_base::in [static]
```


scientific

```
const fmtflags std::ios_base::scientific [static]
```

Generates floating-point output in scientific notation.

Referenced by [std::hexfloat\(\)](#), and [std::scientific\(\)](#).

showbase

```
const fmtflags std::ios_base::showbase [static]
```

Generates a prefix indicating the numeric base of generated integer output.

Referenced by [std::num_put<_CharT, _OutIter>::do_put\(\)](#), [std::noshowbase\(\)](#), and [std::showbase\(\)](#).

showpoint

```
const fmtflags std::ios_base::showpoint [static]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Referenced by [std::noshowpoint\(\)](#), and [std::showpoint\(\)](#).

showpos

```
const fmtflags std::ios_base::showpos [static]
```

Generates a + sign in non-negative generated numeric output.

Referenced by [std::noshowpos\(\)](#), and [std::showpos\(\)](#).

skipws

```
const fmtflags std::ios_base::skipws [static]
```

Skips leading white space before certain input operations.

Referenced by [std::noskipws\(\)](#), [std::__detail::operator>\(\)](#), [std::philox_engine<uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x16565F9A, 0x4E756C83>::operator>\(\)](#), and [std::skipws\(\)](#).

trunc

```
const openmode std::ios_base::trunc [static]
```

Truncate an existing stream when opening. Default for ofstream.

unitbuf

```
const fmtflags std::ios_base::unitbuf [static]
```

Flushes output after each output operation.

Referenced by [std::basic_ostream<_CharT, _Traits>::sentry::~~sentry\(\)](#), [std::nounitbuf\(\)](#), and [std::unitbuf\(\)](#).

uppercase

```
const fmtflags std::ios_base::uppercase [static]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Referenced by [std::num_put<_CharT, _OutIter>::do_put\(\)](#), [std::nouppercase\(\)](#), and [std::uppercase\(\)](#).

The documentation for this class was generated from the following file:

- [ios_base.h](#)

6.555 std::is_abstract<_Tp> Struct Template Reference

```
#include <type_traits>
```

6.555.1 Detailed Description

```
template<typename _Tp>
struct std::is_abstract< _Tp >
```

is_abstract

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.556 std::is_arithmetic< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.556.1 Detailed Description

```
template<typename _Tp>
struct std::is_arithmetic< _Tp >
```

is_arithmetic

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.557 std::is_array< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.557.1 Detailed Description

```
template<typename _Tp>
struct std::is_array< _Tp >
```

is_array

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.558 std::is_assignable< _Tp, _Up > Struct Template Reference

```
#include <type_traits>
```

6.558.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_assignable< _Tp, _Up >
```

is_assignable

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.559 std::is_base_of< _Base, _Derived > Struct Template Reference

```
#include <type_traits>
```

6.559.1 Detailed Description

```
template<typename _Base, typename _Derived>
struct std::is_base_of< _Base, _Derived >
```

`is_base_of`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.560 `std::is_bind_expression< _Tp >` Struct Template Reference

```
#include <functional>
```

6.560.1 Detailed Description

```
template<typename _Tp>
struct std::is_bind_expression< _Tp >
```

Trait that identifies a bind expression.

Determines if the given type `_Tp` is a function object that should be treated as a subexpression when evaluating calls to function objects returned by `std::bind`.

C++11 [func.bind.isbind].

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

6.561 `std::is_bind_expression< _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

6.561.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.562 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

6.562.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.563 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference

```
#include <functional>
```

6.563.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >
```

Class template _Bind is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.564 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference

```
#include <functional>
```

6.564.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
```

Class template _Bind_result is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.565 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference

```
#include <functional>
```

6.565.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const volatile _Bind< _Signature > >
```

Class template _Bind is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.566 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference

```
#include <functional>
```

6.566.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
```

Class template _Bind_result is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.567 `std::is_bind_expression< volatile _Bind< _Signature > >` Struct Template Reference

```
#include <functional>
```

6.567.1 Detailed Description

```
template<typename _Signature>  
struct std::is_bind_expression< volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.568 `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

```
#include <functional>
```

6.568.1 Detailed Description

```
template<typename _Result, typename _Signature>  
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

The documentation for this struct was generated from the following file:

- [functional](#)

6.569 `std::is_class< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.569.1 Detailed Description

```
template<typename _Tp>  
struct std::is_class< _Tp >
```

`is_class`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.570 `std::is_compound< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.570.1 Detailed Description

```
template<typename _Tp>  
struct std::is_compound< _Tp >
```

`is_compound`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.571 `std::is_const<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.571.1 Detailed Description

```
template<typename _Tp>
struct std::is_const<_Tp>
```

`is_const`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.572 `std::is_constructible<_Tp, _Args>` Struct Template Reference

```
#include <type_traits>
```

6.572.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_constructible<_Tp, _Args>
```

`is_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.573 `std::is_copy_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.573.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_assignable<_Tp>
```

`is_copy_assignable`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.574 `std::is_copy_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.574.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_constructible<_Tp>
```

`is_copy_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.575 `std::is_default_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.575.1 Detailed Description

```
template<typename _Tp>
struct std::is_default_constructible< _Tp >
```

`is_default_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.576 `std::is_destructible< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.576.1 Detailed Description

```
template<typename _Tp>
struct std::is_destructible< _Tp >
```

`is_destructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.577 `std::is_empty< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.577.1 Detailed Description

```
template<typename _Tp>
struct std::is_empty< _Tp >
```

`is_empty`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.578 `std::is_enum< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.578.1 Detailed Description

```
template<typename _Tp>
struct std::is_enum< _Tp >
```

`is_enum`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.579 `std::is_error_code_enum< _Tp >` Struct Template Reference

```
#include <system_error>
```

6.579.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_code_enum< _Tp >
```

is_error_code_enum

The documentation for this struct was generated from the following file:

- [system_error](#)

6.580 std::is_error_code_enum< future_errc > Struct Reference

```
#include <future>
```

6.580.1 Detailed Description

Specialization that allows `future_errc` to convert to `error_code`.

The documentation for this struct was generated from the following file:

- [future](#)

6.581 std::is_error_condition_enum< _Tp > Struct Template Reference

```
#include <system_error>
```

6.581.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_condition_enum< _Tp >
```

is_error_condition_enum

The documentation for this struct was generated from the following file:

- [system_error](#)

6.582 std::is_floating_point< _Tp > Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for `std::is_floating_point< _Tp >`:



6.582.1 Detailed Description

```
template<typename _Tp>
struct std::is_floating_point<_Tp>
```

`is_floating_point`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.583 `std::is_function<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.583.1 Detailed Description

```
template<typename _Tp>
struct std::is_function<_Tp>
```

`is_function`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.584 `std::is_fundamental<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.584.1 Detailed Description

```
template<typename _Tp>
struct std::is_fundamental<_Tp>
```

`is_fundamental`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.585 `std::is_integral<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.585.1 Detailed Description

```
template<typename _Tp>
struct std::is_integral<_Tp>
```

`is_integral`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.586 `std::is_layout_compatible<_Tp, _Up>` Struct Template Reference

```
#include <type_traits>
```

6.586.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_layout_compatible< _Tp, _Up >
```

- Remove references and cv-qualifiers.

Since

C++20

C++20

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.587 std::is_literal_type< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.587.1 Detailed Description

```
template<typename _Tp>
struct std::is_literal_type< _Tp >
```

is_literal_type

Deprecated Deprecated in C++17, removed in C++20. The idea of a literal type isn't useful.

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.588 std::is_lvalue_reference< typename > Struct Template Reference

```
#include <type_traits>
```

6.588.1 Detailed Description

```
template<typename>
struct std::is_lvalue_reference< typename >
```

is_lvalue_reference

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.589 std::is_member_function_pointer< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.589.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_function_pointer< _Tp >
```

is_member_function_pointer

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.590 `std::is_member_object_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.590.1 Detailed Description

```
template<typename _Tp>  
struct std::is_member_object_pointer< _Tp >
```

`is_member_object_pointer`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.591 `std::is_member_pointer< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.591.1 Detailed Description

```
template<typename _Tp>  
struct std::is_member_pointer< _Tp >
```

`is_member_pointer`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.592 `std::is_move_assignable< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.592.1 Detailed Description

```
template<typename _Tp>  
struct std::is_move_assignable< _Tp >
```

`is_move_assignable`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.593 `std::is_move_constructible< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.593.1 Detailed Description

```
template<typename _Tp>  
struct std::is_move_constructible< _Tp >
```

`is_move_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.594 `std::is_nothrow_assignable< _Tp, _Up >` Struct Template Reference

```
#include <type_traits>
```

6.594.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >
```

is_nothrow_assignable

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.595 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference

```
#include <type_traits>
```

6.595.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible< _Tp, _Args >
```

is_nothrow_constructible

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.596 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.596.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_assignable< _Tp >
```

is_nothrow_copy_assignable

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.597 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.597.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_constructible< _Tp >
```

is_nothrow_copy_constructible

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.598 std::is_nothrow_default_constructible< _Tp > Struct Template Reference

```
#include <type_traits>
```


6.598.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_default_constructible<_Tp>
```

`is_nothrow_default_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.599 `std::is_nothrow_destructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.599.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_destructible<_Tp>
```

`is_nothrow_destructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.600 `std::is_nothrow_move_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.600.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_assignable<_Tp>
```

`is_nothrow_move_assignable`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.601 `std::is_nothrow_move_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.601.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_constructible<_Tp>
```

`is_nothrow_move_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.602 `std::is_object<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.602.1 Detailed Description

```
template<typename _Tp>
struct std::is_object<_Tp>
```

is_object

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.603 std::is_placeholder<_Tp> Struct Template Reference

```
#include <functional>
```

Inheritance diagram for std::is_placeholder<_Tp>:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

6.603.1 Detailed Description

```
template<typename _Tp>
struct std::is_placeholder<_Tp>
```

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. C++11 [func.bind.isplace].

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

6.604 std::is_placeholder<_Placeholder<_Num>> Struct Template Reference

```
#include <functional>
```

Inheritance diagram for std::is_placeholder<_Placeholder<_Num>>:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

6.604.1 Detailed Description

```
template<int _Num>
```

```
struct std::is_placeholder<_Placeholder<_Num>>
```

Partial specialization of is_placeholder that provides the placeholder number for the placeholder objects defined by libstdc++.

Since

C++11

The documentation for this struct was generated from the following file:

- [functional](#)

6.605 `std::is_pod<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.605.1 Detailed Description

```
template<typename _Tp>
struct std::is_pod<_Tp>
```

`is_pod`

Deprecated Deprecated in C++20. Use `is_standard_layout` && `is_trivial` instead.

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.606 `std::is_pointer<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.606.1 Detailed Description

```
template<typename _Tp>
struct std::is_pointer<_Tp>
```

`is_pointer`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.607 `std::is_pointer_interconvertible_base_of<_Base, _Derived>` Struct Template Reference

```
#include <type_traits>
```

6.607.1 Detailed Description

```
template<typename _Base, typename _Derived>
struct std::is_pointer_interconvertible_base_of<_Base, _Derived>
```

True if `_Derived` is standard-layout and has a base class of type `_Base`.

Since

C++20

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.608 `std::is_polymorphic<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.608.1 Detailed Description

```
template<typename _Tp>
struct std::is_polymorphic< _Tp >
```

`is_polymorphic`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.609 `std::is_reference< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.609.1 Detailed Description

```
template<typename _Tp>
struct std::is_reference< _Tp >
```

`is_reference`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.610 `std::is_rvalue_reference< typename >` Struct Template Reference

```
#include <type_traits>
```

6.610.1 Detailed Description

```
template<typename>
struct std::is_rvalue_reference< typename >
```

`is_rvalue_reference`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.611 `std::is_same< _Tp, _Up >` Struct Template Reference

```
#include <type_traits>
```

6.611.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_same< _Tp, _Up >
```

`is_same`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.612 `std::is_scalar< _Tp >` Struct Template Reference

```
#include <type_traits>
```

6.612.1 Detailed Description

```
template<typename _Tp>
struct std::is_scalar<_Tp >
```

is_scalar

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.613 std::is_signed<_Tp > Struct Template Reference

```
#include <type_traits>
```

6.613.1 Detailed Description

```
template<typename _Tp>
struct std::is_signed<_Tp >
```

is_signed

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.614 std::is_standard_layout<_Tp > Struct Template Reference

```
#include <type_traits>
```

6.614.1 Detailed Description

```
template<typename _Tp>
struct std::is_standard_layout<_Tp >
```

is_standard_layout

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.615 std::is_trivial<_Tp > Struct Template Reference

```
#include <type_traits>
```

6.615.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivial<_Tp >
```

is_trivial

Deprecated Deprecated in C++26. Use a combination of one or more more specialized type traits instead, such as `is_trivially_default_constructible`, `is_trivially_copy_constructible`, `is_trivially_copy_assignable`, etc., depending on the exact check(s) needed.

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.616 std::is_trivially_assignable<_Tp, _Up > Struct Template Reference

```
#include <type_traits>
```

6.616.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_trivially_assignable<_Tp, _Up>
```

`is_trivially_assignable`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.617 `std::is_trivially_constructible<_Tp, _Args>` Struct Template Reference

```
#include <type_traits>
```

6.617.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible<_Tp, _Args>
```

`is_trivially_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.618 `std::is_trivially_copy_assignable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.618.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_assignable<_Tp>
```

`is_trivially_copy_assignable`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.619 `std::is_trivially_copy_constructible<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.619.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_constructible<_Tp>
```

`is_trivially_copy_constructible`

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.620 `std::is_trivially_copyable<_Tp>` Struct Template Reference

```
#include <type_traits>
```

6.620.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copyable< _Tp >
```

is_trivially_copyable

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.621 std::is_trivially_default_constructible< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.621.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_default_constructible< _Tp >
```

is_trivially_default_constructible

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.622 std::is_trivially_destructible< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.622.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_destructible< _Tp >
```

is_trivially_destructible

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.623 std::is_trivially_move_assignable< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.623.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_assignable< _Tp >
```

is_trivially_move_assignable

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.624 std::is_trivially_move_constructible< _Tp > Struct Template Reference

```
#include <type_traits>
```


6.624.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_constructible< _Tp >
```

is_trivially_move_constructible

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.625 std::is_union< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.625.1 Detailed Description

```
template<typename _Tp>
struct std::is_union< _Tp >
```

is_union

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.626 std::is_unsigned< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.626.1 Detailed Description

```
template<typename _Tp>
struct std::is_unsigned< _Tp >
```

is_unsigned

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.627 std::is_void< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.627.1 Detailed Description

```
template<typename _Tp>
struct std::is_void< _Tp >
```

is_void

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.628 std::is_volatile< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.628.1 Detailed Description

```
template<typename _Tp>
struct std::is_volatile< _Tp >
```

is_volatile

The documentation for this struct was generated from the following file:

- [type_traits](#)

6.629 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference

```
#include <stream_iterator.h>
```

Inheritance diagram for std::istream_iterator< _Tp, _CharT, _Traits, _Dist >:



Public Types

- typedef `_CharT` **char_type**
- typedef `ptrdiff_t` [difference_type](#)
- typedef `basic_istream< _CharT, _Traits >` **istream_type**
- typedef `input_iterator_tag` [iterator_category](#)
- typedef `const _Tp *` [pointer](#)
- typedef `const _Tp &` [reference](#)
- typedef `_Traits` **traits_type**
- typedef `_Tp` [value_type](#)

Public Member Functions

- constexpr `istream_iterator` () noexcept(*/*conditional */*)
- constexpr `istream_iterator` (const `istream_iterator` &__obj) noexcept(*/*conditional */*)
- constexpr `istream_iterator` ([default_sentinel_t](#)) noexcept(is_nothrow_default_constructible_v< _Tp >)
- `istream_iterator` (`istream_type` &__s)
- `const _Tp &` **operator*** () const noexcept
- `istream_iterator` & **operator++** ()
- `istream_iterator` **operator++** (int)

- const _Tp * **operator->** () const noexcept
- [istream_iterator](#) & **operator=** (const [istream_iterator](#) &)=default

Friends

- bool **operator==** (const [istream_iterator](#) &__i, [default_sentinel_t](#)) noexcept
- bool **operator==** (const [istream_iterator](#) &__x, const [istream_iterator](#) &__y) noexcept

6.629.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>

class std::istream_iterator<_Tp, _CharT, _Traits, _Dist>

Provides input iterator semantics for streams.

6.629.2 Member Typedef Documentation

difference_type

```
typedef ptrdiff_t std::iterator< input\_iterator\_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::difference_type [inherited]
```

Distance between iterators is represented as this type.

iterator_category

```
typedef input\_iterator\_tag std::iterator< input\_iterator\_tag, _Tp, ptrdiff_t, const _Tp *, const
_Tp & >::iterator_category [inherited]
```

One of the [tag types](#).

pointer

```
typedef const _Tp * std::iterator< input\_iterator\_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::pointer [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef const _Tp & std::iterator< input\_iterator\_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::reference [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef _Tp std::iterator< input\_iterator\_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::←
value_type [inherited]
```

The type "pointed to" by the iterator.

6.629.3 Constructor & Destructor Documentation

istream_iterator() [1/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
```

```
std::istream_iterator<_Tp, _CharT, _Traits, _Dist>::istream_iterator () [inline], [constexpr],
[noexcept]
```

Construct end of input stream iterator.
Referenced by [operator==](#).

istream_iterator() [2/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
    istream_type & __s) [inline]
```

Construct start of input stream iterator.
References [std::__addressof\(\)](#).

6.629.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
bool operator== (
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
    const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [friend]
```

Return true if the iterators refer to the same stream, or are both at end-of-stream.

References [istream_iterator\(\)](#).

The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

6.630 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

```
#include <streambuf_iterator.h>
```

Inheritance diagram for `std::istreambuf_iterator< _CharT, _Traits >`:



Public Types

- typedef `_Traits::off_type` [difference_type](#)

- typedef [input_iterator_tag](#) [iterator_category](#)
- typedef [_CharT](#) [reference](#)
- typedef [_CharT](#) [value_type](#)
- using [pointer](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [_Traits::int_type](#) [int_type](#)
- typedef [basic_streambuf](#)< [_CharT](#), [_Traits](#) > [streambuf_type](#)
- typedef [basic_istream](#)< [_CharT](#), [_Traits](#) > [istream_type](#)

Public Member Functions

- constexpr [istreambuf_iterator](#) () noexcept
- [istreambuf_iterator](#) (const [istreambuf_iterator](#) &) noexcept=default
- constexpr [istreambuf_iterator](#) ([default_sentinel_t](#)) noexcept
- [istreambuf_iterator](#) ([istream_type](#) &__s) noexcept
- [istreambuf_iterator](#) ([streambuf_type](#) *__s) noexcept
- bool [equal](#) (const [istreambuf_iterator](#) &__b) const
- [char_type](#) [operator*](#) () const
- [istreambuf_iterator](#) & [operator++](#) ()
- [istreambuf_iterator](#) [operator++](#) (int)
- [istreambuf_iterator](#) & [operator=](#) (const [istreambuf_iterator](#) &) noexcept=default

Friends

- template<bool _IsMove, typename _CharT2>
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2
([istreambuf_iterator](#)< _CharT2 >, [istreambuf_iterator](#)< _CharT2 >, _CharT2 *)
- template<typename _CharT2, typename _Size>
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_n_a ([istreambuf_iterator](#)< _CharT2 >, _Size, _CharT2 *, bool)
- template<typename _CharT2, typename _Distance>
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type [advance](#) ([istreambuf_iterator](#)< _CharT2 > &, _Distance)
- template<typename _CharT2>
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [ostreambuf_iterator](#)< _CharT2 >::__type [copy](#)
([istreambuf_iterator](#)< _CharT2 >, [istreambuf_iterator](#)< _CharT2 >, [ostreambuf_iterator](#)< _CharT2 >)
- template<typename _CharT2>
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [istreambuf_iterator](#)< _CharT2 >::__type [find](#)
([istreambuf_iterator](#)< _CharT2 >, [istreambuf_iterator](#)< _CharT2 >, const _CharT2 &)
- bool [operator==](#) (const [istreambuf_iterator](#) &__i, [default_sentinel_t](#))

6.630.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >

Provides input iterator semantics for streambufs.

6.630.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
Public typedefs.
```

difference_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_Chart >::difference_type [inherited]
Distance between iterators is represented as this type.
```

int_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
Public typedefs.
```

istream_type

```
template<typename _CharT, typename _Traits>
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
Public typedefs.
```

iterator_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_Chart >::iterator_category [inherited]
One of the tag types.
```

pointer

```
template<typename _CharT, typename _Traits>
using std::istreambuf_iterator< _CharT, _Traits >::pointer
Public typedefs.
```

reference

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT
>::reference [inherited]
This type represents a reference-to-value_type.
```

streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf←
_type
Public typedefs.
```

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
Public typedefs.
```

value_type

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT
>::value_type [inherited]
```

The type "pointed to" by the iterator.

6.630.3 Constructor & Destructor Documentation**istreambuf_iterator() [1/3]**

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator () [inline], [constexpr], [noexcept]
```

Construct end of input stream iterator.

Referenced by [equal\(\)](#), and [operator++\(\)](#).

istreambuf_iterator() [2/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    istream_type & __s) [inline], [noexcept]
```

Construct start of input stream iterator.

istreambuf_iterator() [3/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
    streambuf_type * __s) [inline], [noexcept]
```

Construct start of streambuf iterator.

6.630.4 Member Function Documentation**equal()**

```
template<typename _CharT, typename _Traits>
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
    const istreambuf_iterator< _CharT, _Traits > & __b) const [inline], [nodiscard]
```

Return true both iterators are end or both are not end.

References [istreambuf_iterator\(\)](#).

operator*()

```
template<typename _CharT, typename _Traits>
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* () const [inline], [nodiscard]
```

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator*() on an end of stream is undefined.

operator++() [1/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator & std::istreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

operator++() [2/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
    int ) [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

References [istreambuf_iterator\(\)](#).

The documentation for this class was generated from the following files:

- iosfwd.h
- [streambuf_iterator.h](#)

6.631 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator` Struct Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator`:



Public Types

- typedef `_Alloc::difference_type` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value_type**

Public Member Functions

- **iterator** (`node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0`)
- bool **operator!=** (`const const_iterator &other`) const
- bool **operator!=** (`const iterator &other`) const
- `node_const_pointer` **operator*** () const
- `node_pointer` **operator*** ()
- iterator & **operator++** ()
- iterator **operator++** (int)
- `const node_pointer_pointer` **operator->** () const

- node_pointer_pointer **operator->** ()
- bool **operator==** (const const_iterator &other) const
- bool **operator==** (const iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

6.631.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata >::iterator
```

Child iterator.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

6.632 std::experimental::filesystem::v1::path::iterator Class Reference

```
#include <fs_path.h>
```

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

6.632.1 Detailed Description

An iterator for the components of a path.

The documentation for this class was generated from the following file:

- [experimental/bits/fs_path.h](#)

6.633 `std::filesystem::path::iterator` Class Reference

```
#include <filesystem>
```

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- void **__path_iter_advance** (iterator &__i, difference_type __n) noexcept
- difference_type **__path_iter_distance** (const iterator &__first, const iterator &__last) noexcept
- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

6.633.1 Detailed Description

An iterator for the components of a path.

Since

C++17

The documentation for this class was generated from the following file:

- [bits/fs_path.h](#)

6.634 `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>` Struct Template Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`:



Public Types

- typedef `_Distance` [difference_type](#)
- typedef `_Category` [iterator_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value_type](#)

6.634.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, type-
name _Reference = _Tp&>
```

```
struct std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>
```

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Deprecated Deprecated since C++17. The recommended alternative is to simply define the typedefs directly in your iterator class.

6.634.2 Member Typedef Documentation

`difference_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, type-
name _Reference = _Tp&>
```

```
typedef _Distance std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>::difference_↵
type
```

Distance between iterators is represented as this type.

iterator_category

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_↵  
category
```

One of the [tag types](#).

pointer

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value_type.

reference

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value_type.

value_type

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵  
_Tp*, typename _Reference = _Tp&>  
typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

6.635 std::iterator_traits<_Iterator> Struct Template Reference

6.635.1 Detailed Description

```
template<typename _Iterator>  
struct std::iterator_traits<_Iterator>
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply declares aliases for the nested typedefs from the Iterator argument. Partial specializations for pointers define the typedefs appropriately for the semantics of pointers.

The documentation for this struct was generated from the following file:

- [iterator_concepts.h](#)

6.636 std::iterator_traits<_Tp*> Struct Template Reference

```
#include <stl_iterator_base_types.h>
```

Public Types

- using **difference_type**
- using **iterator_category**
- using **iterator_concept**
- using **pointer**

- using **reference**
- using **value_type**

6.636.1 Detailed Description

```
template<typename _Tp>  
requires is_object_v<_Tp>  
struct std::iterator_traits<_Tp * >
```

Partial specialization for object pointer types.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

6.637 __gnu_pbds::join_error Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for __gnu_pbds::join_error:



Public Member Functions

- virtual const char * [what](#) () const noexcept

6.637.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps).

6.637.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

6.638 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference

```
#include <left_child_next_sibling_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `left_child_next_sibling_heap_const_iterator_< node, _Alloc >` **const_iterator**
- typedef `__rebind_v::const_pointer` **const_pointer**
- typedef `__rebind_v::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `const_iterator` **iterator**
- typedef `left_child_next_sibling_heap_node_< Value_Type, Node_Metadata, _Alloc >` **node**
- typedef `left_child_next_sibling_heap_node_point_const_iterator_< node, _Alloc >` **point_const_iterator**
- typedef `point_const_iterator` **point_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- `left_child_next_sibling_heap` (const Cmp_Fn &)
- `left_child_next_sibling_heap` (const left_child_next_sibling_heap &)
- iterator `begin` ()
- const_iterator `begin` () const
- void `clear` ()
- bool `empty` () const
- iterator `end` ()
- const_iterator `end` () const
- Cmp_Fn & `get_cmp_fn` ()
- const Cmp_Fn & `get_cmp_fn` () const
- size_type `max_size` () const
- size_type `size` () const
- void `swap` (left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef alloc_traits::allocator_type `node_allocator`
- typedef alloc_traits::const_pointer `node_const_pointer`
- typedef Node_Metadata `node_metadata`
- typedef alloc_traits::pointer `node_pointer`
- typedef [std::pair](#)< node_pointer, node_pointer > `node_pointer_pair`

Protected Member Functions

- void `actual_erase_node` (node_pointer)
- void `bubble_to_top` (node_pointer)
- void `clear_imp` (node_pointer)
- node_pointer `get_new_node_for_insert` (const_reference)
- template<typename Pred>
node_pointer `prune` (Pred)
- void `swap_with_parent` (node_pointer, node_pointer)
- void `to_linked_list` ()
- void `value_swap` (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void `make_child_of` (node_pointer, node_pointer)
- static node_pointer `parent` (node_pointer)

Protected Attributes

- node_pointer `m_p_root`
- size_type `m_size`

6.638.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class `__gnu_pbds::detail::left_child_next_sibling_heap`< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Base class for a basic heap.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_.hpp](#)

6.639 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

```
#include <const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

Public Member Functions

- `left_child_next_sibling_heap_const_iterator_()`
- `left_child_next_sibling_heap_const_iterator_(const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other)`
- `left_child_next_sibling_heap_const_iterator_(node_pointer p_nd)`
- `bool operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const`
- `bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`
- `const_reference operator* () const`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & operator++ ()`
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc > operator++ (int)`
- `const_pointer operator-> () const`
- `bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const`
- `bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const`

Public Attributes

- node_pointer `m_p_nd`

6.639.1 Detailed Description

```
template<typename Node, typename _Alloc>
```

```
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

6.639.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_pointer
Iterator's const pointer type.
```

const_reference

```
template<typename Node, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_reference
Iterator's const reference type.
```

difference_type

```
template<typename Node, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
Difference type.
```

iterator_category

```
template<typename Node, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
Category.
```

pointer

```
template<typename Node, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
Iterator's pointer type.
```

reference

```
template<typename Node, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
Iterator's reference type.
```

value_type

```
template<typename Node, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
Iterator's value type.
```

6.639.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
```

```
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↵
next_sibling_heap_const_iterator_ () [inline]
```

Default constructor.

left_child_next_sibling_heap_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↵
next_sibling_heap_const_iterator_ (
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) [inline]
```

Copy constructor.

6.639.4 Member Function Documentation

operator"!="() [1/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content (negatively) to a different iterator object.

operator"!="() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline], [inherited]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

operator==(()) [1/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
    const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content to a different iterator object.

operator==() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator==(
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) const [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/const_iterator.hpp](#)

6.640 `__gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >` Struct Template Reference

```
#include <node.hpp>
```

Public Types

- typedef `_Metadata` **metadata_type**
- typedef [rebind_traits](#)< `_Alloc`, `this_type` >::pointer **node_pointer**
- typedef `_Alloc::size_type` **size_type**
- typedef `_Value` **value_type**

Public Attributes

- `metadata_type` **m_metadata**
- `node_pointer` **m_p_l_child**
- `node_pointer` **m_p_next_sibling**
- `node_pointer` **m_p_prev_or_parent**
- `value_type` **m_value**

6.640.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >
```

Node.

The documentation for this struct was generated from the following file:

- [left_child_next_sibling_heap_/node.hpp](#)

6.641 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<Node, _Alloc >` Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<Node, _Alloc`

>:



Public Types

- typedef [rebind_traits](#)< _Alloc, [value_type](#) >::const_pointer [const_pointer](#)
- typedef [rebind_traits](#)< _Alloc, [value_type](#) >::const_reference [const_reference](#)
- typedef [trivial_iterator_difference_type](#) [difference_type](#)
- typedef [trivial_iterator_tag](#) [iterator_category](#)
- typedef [rebind_traits](#)< _Alloc, [value_type](#) >::pointer [pointer](#)
- typedef [rebind_traits](#)< _Alloc, [value_type](#) >::reference [reference](#)
- typedef Node::value_type [value_type](#)

Public Member Functions

- [left_child_next_sibling_heap_node_point_const_iterator_](#) ()
- [left_child_next_sibling_heap_node_point_const_iterator_](#) (const [left_child_next_sibling_heap_node_point_const_iterator_](#)< Node, _Alloc > &other)
- [left_child_next_sibling_heap_node_point_const_iterator_](#) (node_pointer p_nd)
- bool [operator!=](#) (const [left_child_next_sibling_heap_node_point_const_iterator_](#)< Node, _Alloc > &other) const
- [const_reference operator*](#) () const
- [const_pointer operator->](#) () const
- bool [operator==](#) (const [left_child_next_sibling_heap_node_point_const_iterator_](#)< Node, _Alloc > &other) const

Public Attributes

- node_pointer [m_p_nd](#)

Protected Types

- typedef [rebind_traits](#)< _Alloc, Node >::pointer [node_pointer](#)

6.641.1 Detailed Description

```
template<typename Node, typename _Alloc>  
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

6.641.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>  
typedef rebind\_traits<_Alloc, value\_type>::const_pointer \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::const_pointer  
Iterator's const pointer type.
```

const_reference

```
template<typename Node, typename _Alloc>  
typedef rebind\_traits<_Alloc, value\_type>::const_reference \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::const_reference  
Iterator's const reference type.
```

difference_type

```
template<typename Node, typename _Alloc>  
typedef trivial\_iterator\_difference\_type \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::difference_type  
Difference type.
```

iterator_category

```
template<typename Node, typename _Alloc>  
typedef trivial\_iterator\_tag \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::iterator_category  
Category.
```

pointer

```
template<typename Node, typename _Alloc>  
typedef rebind\_traits<_Alloc, value\_type>::pointer \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::pointer  
Iterator's pointer type.
```

reference

```
template<typename Node, typename _Alloc>  
typedef rebind\_traits<_Alloc, value\_type>::reference \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::reference  
Iterator's reference type.
```

value_type

```
template<typename Node, typename _Alloc>  
typedef Node::value_type \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_  
Node, _Alloc >::value_type  
Iterator's value type.
```

6.641.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_node_point_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::↵
left_child_next_sibling_heap_node_point_const_iterator_ () [inline]
```

Default constructor.

left_child_next_sibling_heap_node_point_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::↵
left_child_next_sibling_heap_node_point_const_iterator_ (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) [inline]
```

Copy constructor.

6.641.4 Member Function Documentation

operator"!="()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) const [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline]
```

Access.

operator==()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
    const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
    other) const [inline]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/point_const_iterator.hpp](#)

6.642 std::length_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::length_error:



Public Member Functions

- **length_error** (const char *)
- **length_error** (const length_error &)=default
- **length_error** (const [string](#) &__arg)
- **length_error** (length_error &&)=default
- length_error & **operator=** (const length_error &)=default
- length_error & **operator=** (length_error &&)=default
- virtual const char * **what** () const noexcept

6.642.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a basic_string instance).

6.642.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept_except.h](#)

6.643 std::less<_Tp> Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less<_Tp>:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

6.643.1 Detailed Description

```
template<typename _Tp>
struct std::less<_Tp>
```

One of the [comparison functors](#).

6.643.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

result_type

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

second_argument_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

6.644 std::ranges::less Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using `is_transparent`

Static Public Member Functions

- template<typename `_Tp`, typename `_Up`>
static constexpr bool **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) noexcept(noexcept(`std::declval`< `_Tp` >())<
`std::declval`< `_Up` >()))

6.644.1 Detailed Description

`ranges::less` function object type.

The documentation for this struct was generated from the following file:

- `ranges_cmp.h`

6.645 std::less< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::less< void >`:



Public Types

- typedef void `first_argument_type`
- typedef `__is_transparent` `is_transparent`
- typedef bool `result_type`
- typedef void `second_argument_type`

Public Member Functions

- template<typename `_Tp`, typename `_Up`>
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`)<
`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)< `std::forward`< `_Up` >(`_u`))
- template<typename `_Tp`, typename `_Up`>
constexpr bool **operator()** (`_Tp` *`_t`, `_Up` *`_u`) const noexcept

6.645.1 Detailed Description

One of the [comparison functors](#).

6.645.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary\_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary\_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary\_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument

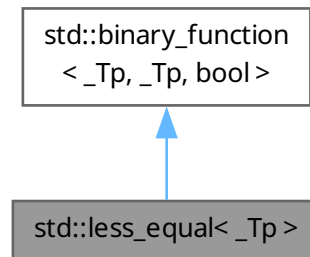
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

6.646 `std::less_equal<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::less_equal<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef bool [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- constexpr bool **operator()** (const `_Tp` &__x, const `_Tp` &__y) const

6.646.1 Detailed Description

```
template<typename _Tp>
struct std::less_equal<_Tp>
```

One of the [comparison functors](#).

6.646.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

result_type

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

second_argument_type

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

6.647 std::ranges::less_equal Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using [is_transparent](#)

Static Public Member Functions

- ```
template<typename _Tp, typename _Up>
static constexpr bool operator() (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval<_Up>())<std::declval<_Tp>()))
```

### 6.647.1 Detailed Description

ranges::less\_equal function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 6.648 std::less\_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::less_equal< void >`:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`_t`)<=[std::forward](#)< `_Up` >(`_u`))) -> decltype([std::forward](#)< `_Tp` >(`_t`)<=[std::forward](#)< `_Up` >(`_u`))
- template<typename `_Tp`, typename `_Up`>  
constexpr bool **operator()** (`_Tp` \*`_t`, `_Up` \*`_u`) const noexcept

#### 6.648.1 Detailed Description

One of the [comparison functors](#).

#### 6.648.2 Member Typedef Documentation

##### `first_argument_type`

typedef void [std::binary\\_function](#)< void, void, bool >::`first_argument_type`  
`first_argument_type` is the type of the first argument

##### `result_type`

typedef bool [std::binary\\_function](#)< void, void, bool >::`result_type`  
`result_type` is the return type

##### `second_argument_type`

typedef void [std::binary\\_function](#)< void, void, bool >::`second_argument_type`  
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.649 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

```
#include <throw_allocator.h>
```

### Public Member Functions

- `limit_adjustor` (const size\_t \_\_l)

#### 6.649.1 Detailed Description

Enter the nth condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.650 `__gnu_cxx::limit_condition` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::limit_condition`:



### Classes

- struct [always\\_adjustor](#)
- struct [limit\\_adjustor](#)
- struct [never\\_adjustor](#)

### Static Public Member Functions

- static size\_t & `count` ()
- static size\_t & `limit` ()
- static void `set_limit` (const size\_t \_\_l)
- static void `throw_conditionally` ()

#### 6.650.1 Detailed Description

Base class for incremental control and throw.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.651 `std::linear_congruential_engine<_UIntType, __a, __c, __m>` Class Template Reference

```
#include <random>
```

### Public Types

- typedef \_UIntType [result\\_type](#)

## Public Member Functions

- [linear\\_congruential\\_engine](#) ()
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
[linear\\_congruential\\_engine](#) (\_Sseq & \_\_q)
- [linear\\_congruential\\_engine](#) (result\_type \_\_s)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) operator() ()
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq> > seed (_Sseq & __q)`
- void [seed](#) (result\_type \_\_s=default\_seed)

## Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

## Static Public Attributes

- static constexpr [result\\_type](#) [default\\_seed](#)
- static constexpr [result\\_type](#) [increment](#)
- static constexpr [result\\_type](#) [modulus](#)
- static constexpr [result\\_type](#) [multiplier](#)

## Friends

- `template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>`  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os, const  
[std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > & \_\_lcr)
- bool [operator==](#) (const [linear\\_congruential\\_engine](#) & \_\_lhs, const [linear\\_congruential\\_engine](#) & \_\_rhs)
- `template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>`  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > & \_\_is,  
[std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > & \_\_lcr)

### 6.651.1 Detailed Description

`template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`  
**class** [std::linear\\_congruential\\_engine](#)< \_UIntType, \_\_a, \_\_c, \_\_m >

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Since

C++11

### 6.651.2 Member Typedef Documentation

#### result\_type

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

### 6.651.3 Constructor & Destructor Documentation

#### linear\_congruential\_engine() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine () [inline]
```

Constructs a linear\_congruential\_engine random number generator engine with seed 1.

#### linear\_congruential\_engine() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
 result_type __s) [inline], [explicit]
```

Constructs a linear\_congruential\_engine random number generator engine with seed \_\_s. The default seed value is 1.

#### Parameters

|                   |                         |
|-------------------|-------------------------|
| $\leftrightarrow$ | The initial seed value. |
| <code>__s</code>  |                         |

#### linear\_congruential\_engine() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
 _Sseq & __q) [inline], [explicit]
```

Constructs a linear\_congruential\_engine random number generator engine seeded from the seed sequence \_\_q.

#### Parameters

|                   |                    |
|-------------------|--------------------|
| $\leftrightarrow$ | the seed sequence. |
| <code>__q</code>  |                    |

### 6.651.4 Member Function Documentation

#### discard()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

#### max()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max () [inline],
```

[static], [constexpr]

Gets the largest possible value in the output range.

### min()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::min () [inline],
[static], [constexpr]
```

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

### operator>()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::operator() () [inline]
```

Gets the next random number in the sequence.

### seed() [1/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
auto std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed (
 _Sseq & __q) -> _If_seed_seq<_Sseq >
```

Reseeds the `linear_congruential_engine` random number generator engine sequence using values from the seed sequence `__q`.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Seeds the LCR engine with a value generated by `__q`.

References [std::\\_\\_lg\(\)](#), [std::\\_\\_numeric\\_limits\\_base::digits](#), and [seed\(\)](#).

### seed() [2/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed (
 result_type __s = default_seed)
```

Reseeds the `linear_congruential_engine` random number generator engine sequence to the seed `__s`.

#### Parameters

|                  |               |
|------------------|---------------|
| <code>__s</code> | The new seed. |
|------------------|---------------|

Seeds the LCR with integral value `__s`, adjusted so that the ring identity is never a member of the convergence set.

Referenced by [std::linear\\_congruential\\_engine< uint\\_fast32\\_t, 16807UL, 0UL, 2147483647UL >::linear\\_congruential\\_engine\(\)](#), [std::linear\\_congruential\\_engine< uint\\_fast32\\_t, 16807UL, 0UL, 2147483647UL >::linear\\_congruential\\_engine\(\)](#), and [seed\(\)](#).



### 6.651.5 Friends And Related Symbol Documentation

#### `operator<<`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr) [friend]
```

Writes the textual representation of the state  $x(i)$  of  $x$  to `__os`.

#### Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <code>__os</code>  | The output stream.                                                   |
| <code>__lcr</code> | A % <code>linear_congruential_engine</code> random number generator. |

#### Returns

`__os`.

#### `operator==`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
 const linear_congruential_engine<_UIntType, __a, __c, __m> & __lhs,
 const linear_congruential_engine<_UIntType, __a, __c, __m> & __rhs) [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

#### Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__lhs</code> | A linear congruential random number generator object.       |
| <code>__rhs</code> | Another linear congruential random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

#### `operator>>`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr) [friend]
```

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

**Parameters**

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <code>__is</code>  | The input stream.                                                    |
| <code>__lcr</code> | A % <code>linear_congruential_engine</code> random number generator. |

**Returns**

`__is`.

**6.651.6 Member Data Documentation****increment**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment [static],
[constexpr]
```

An increment.

**modulus**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus [static], [constexpr]
```

The modulus.

**multiplier**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier [static],
[constexpr]
```

The multiplier.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.652 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference**

```
#include <hash_policy.hpp>
```

**Public Types**

- typedef `Size_Type` **size\_type**

**Public Member Functions**

- void **swap** ([linear\\_probe\\_fn](#)< `Size_Type` > &other)

**Protected Member Functions**

- `size_type` [operator\(\)](#) (`size_type` i) const

### 6.652.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

### 6.652.2 Member Function Documentation

#### operator()()

```
template<typename Size_Type>
linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 6.653 std::\_\_debug::list< \_Tp, \_Allocator > Class Template Reference

```
#include <list>
```

Inheritance diagram for std::\_\_debug::list< \_Tp, \_Allocator >:



### Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, list > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, list > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **list** (`_Base_ref __x`)
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
**list** (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- **list** (`const _Allocator &__a`) `noexcept`
- **list** (`const list &`)=`default`
- **list** (`const list &__x, const __type_identity_t< allocator_type > &__a`)
- **list** (`initializer_list< value_type > __l, const allocator_type &__a=allocator_type()`)
- **list** (`list &&`)=`default`
- **list** (`list &&__x, const __type_identity_t< allocator_type > &__a`) `noexcept`(`std::is_nothrow_constructible< _↔_Base, _Base, const allocator_type & >::value`)
- **list** (`size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator()`)
- **list** (`size_type __n, const allocator_type &__a=allocator_type()`)
- `const _Base & M_base () const noexcept`
- `_Base & M_base () noexcept`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
**void assign** (`_InputIterator __first, _InputIterator __last`)
- **void assign** (`initializer_list< value_type > __l`)
- **void assign** (`size_type __n, const _Tp &__t`)
- `const_reference back () const noexcept`
- `reference back () noexcept`
- `const_iterator begin () const noexcept`
- `iterator begin () noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- **void clear** () `noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
**iterator emplace** (`const_iterator __position, _Args &&... __args`)
- `const_iterator end () const noexcept`
- `iterator end () noexcept`
- `iterator erase (const_iterator __first, const_iterator __last) noexcept`
- `iterator erase (const_iterator __position) noexcept`
- `const_reference front () const noexcept`
- `reference front () noexcept`
- `iterator insert (const_iterator __p, initializer_list< value_type > __l)`
- `template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
**iterator insert** (`const_iterator __position, _InputIterator __first, _InputIterator __last`)
- `iterator insert (const_iterator __position, _Tp &&__x)`
- `iterator insert (const_iterator __position, const _Tp &__x)`
- `iterator insert (const_iterator __position, size_type __n, const _Tp &__x)`
- **void merge** (`list &&__x`)
- `template<class _Compare>`  
**void merge** (`list &&__x, _Compare __comp`)
- **void merge** (`list &__x`)
- `template<typename _Compare>`  
**void merge** (`list &__x, _Compare __comp`)
- `list & operator= (const list &)=default`
- `list & operator= (initializer_list< value_type > __l)`
- `list & operator= (list &&)=default`

- void **pop\_back** () noexcept
- void **pop\_front** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- \_\_remove\_return\_type **remove** (const \_Tp &\_\_value)
- template<class \_Predicate>  
\_\_remove\_return\_type **remove\_if** (\_Predicate \_\_pred)
- const\_reverse\_iterator **rend** () const noexcept
- reverse\_iterator **rend** () noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **sort** ()
- template<typename \_StrictWeakOrdering>  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **splice** (const\_iterator \_\_position, list &&\_\_x) noexcept
- void **splice** (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void **splice** (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_i) noexcept
- void **splice** (const\_iterator \_\_position, list &\_\_x) noexcept
- void **splice** (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void **splice** (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_i) noexcept
- void **swap** (list &\_\_x) noexcept(*/\*conditional \*/*)
- \_\_remove\_return\_type **unique** ()
- template<class \_BinaryPredicate>  
\_\_remove\_return\_type **unique** (\_BinaryPredicate \_\_binary\_pred)

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_\_Safe\_iterator

#### 6.653.1 Detailed Description

**template<typename \_Tp, typename \_Allocator = std::allocator<\_Tp>>**  
**class std::\_\_debug::list< \_Tp, \_Allocator >**

Class std::list with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/list](#)

## 6.654 std::list< \_Tp, \_Alloc > Class Template Reference

```
#include <stl_list.h>
```

Inheritance diagram for `std::list<_Tp, _Alloc>`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Node_traits::_Const_iterator` **const\_iterator**
- typedef `_Tp_alloc_traits::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Node_traits::_Iterator` **iterator**
- typedef `_Tp_alloc_traits::pointer` **pointer**
- typedef `_Tp_alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- `list()`=default
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
`list` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `const allocator_type &__a=allocator_type()`)
- `list` (`const allocator_type &__a`) noexcept
- `list` (`const list &__x`)
- `list` (`const list &__x`, `const __type_identity_t< allocator_type > &__a`)
- `list` (`initializer_list< value_type > __l`, `const allocator_type &__a=allocator_type()`)
- `list` (`list &&`)=default
- `list` (`list &&__x`, `const __type_identity_t< allocator_type > &__a`) noexcept(`_Node_alloc_traits::_S_always_equal()`)
- `list` (`size_type __n`, `const allocator_type &__a=allocator_type()`)
- `list` (`size_type __n`, `const value_type &__value`, `const allocator_type &__a=allocator_type()`)
- `~list` ()=default
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
void `assign` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void `assign` (`initializer_list< value_type > __l`)
- void `assign` (`size_type __n`, `const value_type &__val`)

- const\_reference [back](#) () const noexcept
- reference [back](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- iterator [begin](#) () noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- template<typename... \_Args>  
[iterator](#) [emplace](#) (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
reference [emplace\\_back](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
reference [emplace\\_front](#) (\_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- const\_iterator [end](#) () const noexcept
- iterator [end](#) () noexcept
- iterator [erase](#) (const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- [iterator](#) [erase](#) (const\_iterator \_\_position) noexcept
- const\_reference [front](#) () const noexcept
- reference [front](#) () noexcept
- allocator\_type [get\\_allocator](#) () const noexcept
- iterator [insert](#) (const\_iterator \_\_p, [initializer\\_list](#)<value\_type> \_\_l)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
[iterator](#) [insert](#) (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [iterator](#) [insert](#) (const\_iterator \_\_position, [size\\_type](#) \_\_n, const [value\\_type](#) &\_\_x)
- [size\\_type](#) [max\\_size](#) () const noexcept
- void [merge](#) (list &&\_\_x)
- template<typename \_StrictWeakOrdering>  
void [merge](#) (list &&\_\_x, \_StrictWeakOrdering \_\_comp)
- void [merge](#) (list &&\_\_x)
- template<typename \_StrictWeakOrdering>  
void [merge](#) (list &&\_\_x, \_StrictWeakOrdering \_\_comp)
- list & [operator=](#) (const list &&\_\_x)
- list & [operator=](#) ([initializer\\_list](#)<value\_type> \_\_l)
- list & [operator=](#) (list &&\_\_x) noexcept(\_Node\_alloc\_traits::\_S\_nothrow\_move())
- void [pop\\_back](#) () noexcept
- void [pop\\_front](#) () noexcept
- void [push\\_back](#) (const value\_type &\_\_x)
- void [push\\_back](#) (value\_type &&\_\_x)
- void [push\\_front](#) (const value\_type &\_\_x)
- void [push\\_front](#) (value\_type &&\_\_x)
- const\_reverse\_iterator [rbegin](#) () const noexcept
- reverse\_iterator [rbegin](#) () noexcept
- \_\_remove\_return\_type [remove](#) (const \_Tp &\_\_value)
- template<typename \_Predicate>  
\_\_remove\_return\_type [remove\\_if](#) (\_Predicate)
- const\_reverse\_iterator [rend](#) () const noexcept
- reverse\_iterator [rend](#) () noexcept
- void [resize](#) ([size\\_type](#) \_\_new\_size)

- void `resize` (`size_type` \_\_new\_size, const `value_type` &\_\_x)
- void `reverse` () noexcept
- `size_type` `size` () const noexcept
- void `sort` ()
- template<typename \_StrictWeakOrdering>  
void `sort` (\_StrictWeakOrdering)
- void `splice` (const\_iterator \_\_position, `list` &&\_\_x) noexcept
- void `splice` (const\_iterator \_\_position, `list` &&\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void `splice` (const\_iterator \_\_position, `list` &\_\_x) noexcept
- void `splice` (const\_iterator \_\_position, `list` &\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void `swap` (`list` &\_\_x) noexcept
- \_\_remove\_return\_type `unique` ()
- template<typename \_BinaryPredicate>  
\_\_remove\_return\_type `unique` (\_BinaryPredicate)
- `iterator` `insert` (const\_iterator \_\_position, const `value_type` &\_\_x)
- `iterator` `insert` (const\_iterator \_\_position, `value_type` &&\_\_x)
- void `splice` (const\_iterator \_\_position, `list` &&\_\_x, const\_iterator \_\_i) noexcept
- void `splice` (const\_iterator \_\_position, `list` &\_\_x, const\_iterator \_\_i) noexcept

## Protected Types

- typedef `_Node_alloc_traits::pointer` `_Node_ptr`

## Protected Member Functions

- template<typename \_InputIterator>  
void `_M_assign_dispatch` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
void `_M_assign_dispatch` (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void `_M_check_equal_allocators` (const `list` &\_\_x) noexcept
- void `_M_clear` () noexcept
- template<typename... \_Args>  
\_Node\_ptr `_M_create_node` (\_Args &&... \_\_args)
- void `_M_dec_size` (size\_t \_\_n)
- void `_M_default_append` (`size_type` \_\_n)
- void `_M_default_initialize` (size\_type \_\_n)
- void `_M_destroy_node` (\_Node\_ptr \_\_p)
- size\_t `_M_distance` (const \_\_detail::List\_node\_base \* \_\_first, const \_\_detail::List\_node\_base \* \_\_last) const
- void `_M_erase` (iterator \_\_position) noexcept
- void `_M_fill_assign` (`size_type` \_\_n, const `value_type` &\_\_val)
- void `_M_fill_initialize` (size\_type \_\_n, const `value_type` &\_\_x)
- `_Node_alloc_traits::pointer` `_M_get_node` ()
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const noexcept
- `_Node_alloc_type` & `_M_get_Node_allocator` () noexcept
- size\_t `_M_get_size` () const
- void `_M_inc_size` (size\_t \_\_n)
- void `_M_init` () noexcept



- `template<typename _InputIterator>`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _Integer>`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename... _Args>`  
`void _M_insert (iterator __position, _Args &&... __args)`
- `void _M_move_assign (list &&__x, false_type)`
- `void _M_move_assign (list &&__x, true_type) noexcept`
- `void _M_move_nodes (_List_base &&__x)`
- `size_t _M_node_count () const`
- `void _M_put_node (_Node_ptr __p) noexcept`
- `const_iterator _M_resize_pos (size_type &__new_size) const`
- `void _M_set_size (size_t __n)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

### Static Protected Member Functions

- `static size_t _S_distance (const __detail:: List_node_base * __first, const __detail:: List_node_base * __last)`
- `static size_t _S_distance (const_iterator __first, const_iterator __last)`

### Protected Attributes

- `_List_impl _M_impl`

#### 6.654.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`  
`class std::list< _Tp, _Alloc >`

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

A <----> B <----> C <----> D

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

## 6.654.2 Constructor & Destructor Documentation

### list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list () [default]
```

Creates a list with no elements.

Referenced by [erase\(\)](#), [insert\(\)](#), [insert\(\)](#), [insert\(\)](#), [merge\(\)](#), [merge\(\)](#), [operator=\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

### list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 const allocator_type & __a) [inline], [explicit], [noexcept]
```

Creates a list with no elements.

#### Parameters

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__a</code> | An allocator object. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

### list() [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a list with default constructed elements.

#### Parameters

|                                                                                                                         |                                             |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| <a href="#"></a><br><code>__n</code> | The number of elements to initially create. |
| <a href="#"></a><br><code>__a</code> | An allocator object.                        |

This constructor fills the list with `__n` default constructed elements.

### list() [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline]
```

Creates a list with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

This constructor fills the list with `__n` copies of `__value`.

#### list() [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
 const list<_Tp, _Alloc> & __x) [inline]
```

List copy constructor.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The newly-created list uses a copy of the allocation object used by `__x` (unless the allocator traits dictate a different object).

#### list() [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
 list<_Tp, _Alloc> &&) [default]
```

List move constructor.

The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

#### list() [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (
 initializer_list<value_type> & __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from an `initializer_list`.

#### Parameters

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
| <code>__a</code> | An allocator object.                                          |

Create a list consisting of copies of the elements in the `initializer_list` `__l`. This is linear in `__l.size()`.

#### list() [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
```

```
std::list< _Tp, _Alloc >::list (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from a range.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a list consisting of copies of the elements from `[__first, __last)`. This is linear in N (where N is distance(`__first`, `__last`)).

#### `~list()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~list () [default]
```

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### 6.654.3 Member Function Documentation

#### `_M_create_node()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
_Node_ptr std::list< _Tp, _Alloc >::_M_create_node (
 _Args &&... __args) [inline], [protected]
```

#### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__args</code> | An instance of user data. |
|---------------------|---------------------------|

Allocates space for a new node and constructs a copy of `__args` in it.

Referenced by [emplace\(\)](#), and [insert\(\)](#).

#### `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::list< _Tp, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a list.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a list with copies of the elements in the range [*\_\_first*,*\_\_last*).

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

### assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::assign (
 initializer_list< value_type > __l) [inline]
```

Assigns an initializer\_list to a list.

#### Parameters

|   |                                    |
|---|------------------------------------|
| ↩ | An initializer_list of value_type. |
| ↩ |                                    |
| ↩ |                                    |
| ↩ |                                    |
| / |                                    |

Replace the contents of the list with copies of the elements in the initializer\_list \_\_l. This is linear in \_\_l.size().

### assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::assign (
 size_type __n,
 const value_type & __val) [inline]
```

Assigns a given value to a list.

#### Parameters

|       |                                    |
|-------|------------------------------------|
| __n   | Number of elements to be assigned. |
| __val | Value to be assigned.              |

This function fills a list with \_\_n copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

### back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list<_Tp, _Alloc>::back () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

### back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list<_Tp, _Alloc>::back () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

### begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::begin () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

### **begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::begin\(\) [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Referenced by [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::list\(\)](#), [insert\(\)](#), [insert\(\)](#), [merge\(\)](#), [merge\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::operator=\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::unique\(\)](#), and [unique\(\)](#).

### **cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::cbegin\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

### **cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::cend\(\) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

### **clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::clear\(\) [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by [operator=\(\)](#).

### **crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crbegin\(\) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

### **crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list<_Tp, _Alloc>::crend\(\) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

### **emplace()**

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
list<_Tp, _Alloc>::iterator list::emplace (
 const_iterator __position,
 _Args &&... __args)
```

Constructs object in list before specified iterator.

#### Parameters

|                         |                                 |
|-------------------------|---------------------------------|
| <code>__position</code> | A const_iterator into the list. |
| <code>__args</code>     | Arguments.                      |

#### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with `T(std::forward<Args>(args)...) before the specified location`. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references. References [\\_M\\_create\\_node\(\)](#), and [std::forward\(\)](#).

#### `empty()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list<_Tp, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Referenced by [insert\(\)](#), and [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::splice\(\)](#).

#### `end()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::end () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

#### `end()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::end () [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Referenced by [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::list\(\)](#), [merge\(\)](#), [merge\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::operator=\(\)](#), and [std::list<\\_Tp, polymorphic\\_allocator<\\_Tp>>::splice\(\)](#).

#### `erase()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::erase (
 const_iterator __first,
 const_iterator __last) [inline], [noexcept]
```

Remove a range of elements.

#### Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::erase (
 const_iterator __position) [noexcept]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), and [erase\(\)](#).

Referenced by [erase\(\)](#), [resize\(\)](#), and [resize\(\)](#).

**front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

**front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front () [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

**get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Referenced by [insert\(\)](#), [insert\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

**insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
 const_iterator __p,
 initializer_list< value_type > __l) [inline]
```



Inserts the contents of an initializer\_list into list before specified const\_iterator.

#### Parameters

|                                  |                                    |
|----------------------------------|------------------------------------|
| <a href="#"><code>__p</code></a> | A const_iterator into the list.    |
| <a href="#"><code>__l</code></a> | An initializer_list of value_type. |

#### Returns

An iterator pointing to the first element inserted (or `__p`).

This function will insert copies of the data in the initializer\_list `__l` into the list before the location specified by `__p`. This operation is linear in the number of elements inserted and does not invalidate iterators and references.

#### insert() [2/5]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
list<_Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last)
```

Inserts a range into the list.

#### Parameters

|                                         |                            |
|-----------------------------------------|----------------------------|
| <a href="#"><code>__position</code></a> | An iterator into the list. |
| <a href="#"><code>__first</code></a>    | An input iterator.         |
| <a href="#"><code>__last</code></a>     | An input iterator.         |

#### Returns

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert copies of the data in the range `[__first, __last)` into the list before the location specified by `__position`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

References [list\(\)](#), [begin\(\)](#), [empty\(\)](#), [get\\_allocator\(\)](#), and [splice\(\)](#).

#### insert() [3/5]

```
template<typename _Tp, typename _Alloc>
list<_Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 const value_type & __x)
```

Inserts given value into list before specified iterator.

**Parameters**

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
| <code>__x</code>        | Data to be inserted.       |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

References [list\(\)](#), [\\_M\\_create\\_node\(\)](#), and [insert\(\)](#).

Referenced by [insert\(\)](#), and [resize\(\)](#).

**insert()** [4/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x)
```

Inserts a number of copies of given data into the list.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the list.         |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

**Returns**

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

References [list\(\)](#), [begin\(\)](#), [get\\_allocator\(\)](#), and [splice\(\)](#).

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts given value into list before specified iterator.

**Parameters**

|                         |                            |
|-------------------------|----------------------------|
| <code>__position</code> | An iterator into the list. |
|-------------------------|----------------------------|

|                  |                      |
|------------------|----------------------|
| <code>__x</code> | Data to be inserted. |
|------------------|----------------------|

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list<_Tp, _Alloc>::max_size () const [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible list.

**merge()** [1/2]

```
template<typename _Tp, typename _Alloc>
void list::merge (
 list<_Tp, _Alloc> && __x)
```

Merge sorted lists.

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <code>__x</code> | Sorted list to merge. |
|------------------|-----------------------|

Assumes that both `__x` and this list are sorted according to `operator<()`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [end\(\)](#), and [merge\(\)](#).

Referenced by [merge\(\)](#), and [merge\(\)](#).

**merge()** [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::merge (
 list<_Tp, _Alloc> && __x,
 _StrictWeakOrdering __comp)
```

Merge sorted lists according to comparison function.

**Template Parameters**

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <code>_StrictWeakOrdering</code> | Comparison function defining sort order. |
|----------------------------------|------------------------------------------|

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__x</code>    | Sorted list to merge. |
| <code>__comp</code> | Comparison functor.   |

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [end\(\)](#), and [merge\(\)](#).

### **operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc > & list::operator= (
 const list< _Tp, _Alloc > & __x)
```

List assignment operator.

#### **Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

All the elements of `__x` are copied.

Whether the allocator is copied depends on the allocator traits.

References [list\(\)](#), [std::\\_\\_addressof\(\)](#), [begin\(\)](#), [clear\(\)](#), and [end\(\)](#).

### **operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

List initializer list assignment operator.

#### **Parameters**

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An initializer_list of value_type. |
|------------------|------------------------------------|

Replace the contents of the list with copies of the elements in the initializer\_list `__l`. This is linear in `l.size()`.

### **operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
 list< _Tp, _Alloc > && __x) [inline], [noexcept]
```

List move assignment operator.

#### **Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The contents of `__x` are moved into this list (without copying).

Afterwards `__x` is a valid, but unspecified list

Whether the allocator is moved depends on the allocator traits.

**pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

**pop\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop\_front() is called.

**push\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_back (
 const value_type & __x) [inline]
```

Add data to the end of the list.

**Parameters**

|                         |                   |
|-------------------------|-------------------|
| <b><code>__x</code></b> | Data to be added. |
|-------------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

**push\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
 const value_type & __x) [inline]
```

Add data to the front of the list.

**Parameters**

|                         |                   |
|-------------------------|-------------------|
| <b><code>__x</code></b> | Data to be added. |
|-------------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

`const_reverse_iterator` [std::list](#)< `_Tp`, `_Alloc` >::rbegin () const [inline], [nodiscard], [noexcept]  
Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

### rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin () [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

### remove()

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::remove (
 const _Tp & __value)
```

Remove all elements equal to `value`.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__value</code> | The value to remove. |
|----------------------|----------------------|

Removes every element in the list equal to `value`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

### remove\_if()

```
template<typename _Tp, typename _Alloc>
template<typename _Predicate>
list< _Tp, _Alloc >::__remove_return_type list::remove_if (
 _Predicate __pred)
```

Remove all elements satisfying a predicate.

#### Template Parameters

|                         |                                     |
|-------------------------|-------------------------------------|
| <code>_Predicate</code> | Unary predicate function or object. |
|-------------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

### rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

### rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
reverse_iterator std::list<_Tp, _Alloc>::rend() [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

### resize() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
 size_type __new_size)
```

Resizes the list to the specified number of elements.

#### Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain. |
|-------------------------|---------------------------------------------|

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

References [std::end\(\)](#), and [erase\(\)](#).

### resize() [2/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
 size_type __new_size,
 const value_type & __x)
```

Resizes the list to the specified number of elements.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain.       |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

References [std::end\(\)](#), [erase\(\)](#), and [insert\(\)](#).

### reverse()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::reverse() [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

### size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list<_Tp, _Alloc>::size() const [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the list.

Referenced by [std::operator<=>\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

### sort() [1/2]

```
template<typename _Tp, typename _Alloc>
```

```
void list::sort ()
```

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [std::begin\(\)](#), [std::empty\(\)](#), and [std::end\(\)](#).

### sort() [2/2]

```
template<typename _Tp, typename _Alloc>
```

```
template<typename _StrictWeakOrdering>
```

```
void list::sort (
```

```
 _StrictWeakOrdering __comp)
```

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References [std::begin\(\)](#), [std::empty\(\)](#), and [std::end\(\)](#).

### splice() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
void std::list< _Tp, _Alloc >::splice (
```

```
 const_iterator __position,
```

```
 list< _Tp, _Alloc > && __x) [inline], [noexcept]
```

Insert contents of another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this `!= __x`.

Referenced by [insert\(\)](#), [insert\(\)](#), [remove\(\)](#), [remove\\_if\(\)](#), [unique\(\)](#), and [unique\(\)](#).

### splice() [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
void std::list< _Tp, _Alloc >::splice (
```

```
 const_iterator __position,
```

```
 list< _Tp, _Alloc > && __x,
```

```
 const_iterator __first,
```

```
 const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__first</code>    | Iterator referencing the start of range in x.      |
| <code>__last</code>     | Iterator referencing the end of range in x.        |



Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time. Undefined if `__position` is in [`__first`,`__last`).

**splice()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > && __x,
 const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

**Parameters**

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__i</code>        | Iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

**splice()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > & __x,
 const_iterator __first,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

**Parameters**

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__first</code>    | Const_iterator referencing the start of range in x.      |
| <code>__last</code>     | Const_iterator referencing the end of range in x.        |

Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time. Undefined if `__position` is in [`__first`,`__last`).

**splice()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
```

```
list< _Tp, _Alloc > & __x,
const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

#### Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |
| <code>__i</code>        | Iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

#### swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
 list< _Tp, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another list.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A list of the same element and allocator types. |
|------------------|-------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

#### unique() [1/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::unique ()
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

#### unique() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinaryPredicate>
list< _Tp, _Alloc >::__remove_return_type list::unique (
 _BinaryPredicate __binary_pred)
```

Remove consecutive elements satisfying a predicate.

### Template Parameters

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>_BinaryPredicate</code> | Binary predicate function or object. |
|-------------------------------|--------------------------------------|

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

References [list\(\)](#), [std::begin\(\)](#), [begin\(\)](#), [std::end\(\)](#), [get\\_allocator\(\)](#), [size\(\)](#), and [splice\(\)](#).

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

## 6.655 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

### Public Types

- typedef `list_update_tag` `container_category`
- typedef `Eq_Fn` `eq_fn`
- typedef `Update_Policy` `update_policy`

### Public Member Functions

- `list_update` (`const list_update &other`)
- template<typename `It`>  
  `list_update` (`It first, It last`)
- `list_update & operator=` (`const list_update &other`)
- void `swap` (`list_update &other`)

#### 6.655.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class
Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
```

A list-update based associative container.

### Template Parameters

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                         |
| <i>Mapped</i>        | Map type.                                                                         |
| <i>Eq_Fn</i>         | Equal functor.                                                                    |
| <i>Update_Policy</i> | Update policy, determines when an element will be moved to the front of the list. |
| <i>_Alloc</i>        | Allocator type.                                                                   |

Base is `detail::lu_map`.

## 6.655.2 Constructor & Destructor Documentation

### list\_update()

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type,
class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
 It first,
 It last) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 6.656 \_\_gnu\_pbds::list\_update\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::list\_update\_tag:



### 6.656.1 Detailed Description

List-update.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.657 std::locale Class Reference

```
#include <locale_classes.h>
```

### Classes

- class [facet](#)
- class [id](#)

## Public Types

- typedef int [category](#)

## Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const char \*\_\_s)
- [locale](#) (const locale &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const locale &\_\_base, const locale &\_\_add, [category](#) \_\_cat)
- [locale](#) (const locale &\_\_base, const [std::string](#) &\_\_s, [category](#) \_\_cat)
- [locale](#) (const locale &\_\_other) throw ()
- template<typename \_Facet>  
[locale](#) (const locale &\_\_other, \_Facet \*\_\_f)
- [locale](#) (const [std::string](#) &\_\_s)
- [~locale](#) () throw ()
- template<typename \_Facet>  
[locale combine](#) (const [locale](#) &\_\_other) const
- [string name](#) () const
- template<typename \_Char, typename \_Traits, typename \_Alloc>  
bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s2) const
- template<typename \_CharT, typename \_Traits, typename \_Alloc>  
bool [operator\(\)](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &\_\_other) throw ()
- bool [operator==](#) (const [locale](#) &\_\_other) const throw ()

## Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale global](#) (const [locale](#) &\_\_loc)

## Static Public Attributes

- static const [category](#) [none](#)
- static const [category](#) [ctype](#)
- static const [category](#) [numeric](#)
- static const [category](#) [collate](#)
- static const [category](#) [time](#)
- static const [category](#) [monetary](#)
- static const [category](#) [messages](#)
- static const [category](#) [all](#)

## Friends

- template<typename \_Facet>  
const \_Facet \* [\\_\\_try\\_use\\_facet](#) (const [locale](#) &\_\_loc)
- template<typename \_Cache>  
struct [\\_\\_use\\_cache](#)
- class [\\_Impl](#)
- class [facet](#)

- `template<typename _Facet>`  
`bool has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet>`  
`const _Facet & use\_facet (const locale &__loc)`

### 6.657.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

### 6.657.2 Member Typedef Documentation

#### category

```
typedef int std::locale::category
```

Definition of `locale::category`.

### 6.657.3 Constructor & Destructor Documentation

#### `locale()` [1/8]

```
std::locale::locale () throw ()
```

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [locale\(\)](#), [~locale\(\)](#), [combine\(\)](#), [global\(\)](#), [has\\_facet](#), [operator\(\)](#), [operator=\(\)](#), [operator==\(\)](#), and [use\\_facet](#).

#### `locale()` [2/8]

```
std::locale::locale (
 const locale & __other) throw ()
```

Copy constructor.

Constructs a copy of *other*.

#### Parameters

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

References [locale\(\)](#).

#### `locale()` [3/8]

```
std::locale::locale (
 const char * __s) [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

#### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

**Exceptions**

|                                                 |                                                     |
|-------------------------------------------------|-----------------------------------------------------|
| <a href="#"><code>std::runtime_error</code></a> | if <code>__s</code> is null or an undefined locale. |
|-------------------------------------------------|-----------------------------------------------------|

References [locale\(\)](#).

**locale()** [4/8]

```
std::locale::locale (
 const locale & __base,
 const char * __s,
 category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

**Exceptions**

|                                                 |                                                     |
|-------------------------------------------------|-----------------------------------------------------|
| <a href="#"><code>std::runtime_error</code></a> | if <code>__s</code> is null or an undefined locale. |
|-------------------------------------------------|-----------------------------------------------------|

References [locale\(\)](#).

**locale()** [5/8]

```
std::locale::locale (
 const std::string & __s) [inline], [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

**Exceptions**

|                                                 |                                             |
|-------------------------------------------------|---------------------------------------------|
| <a href="#"><code>std::runtime_error</code></a> | if <code>__s</code> is an undefined locale. |
|-------------------------------------------------|---------------------------------------------|

References [locale\(\)](#).

**locale()** [6/8]

```
std::locale::locale (
```

```
const locale & __base,
const std::string & __s,
category __cat) [inline]
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

#### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

#### Exceptions

|                                 |                                             |
|---------------------------------|---------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is an undefined locale. |
|---------------------------------|---------------------------------------------|

References [locale\(\)](#).

#### locale() [7/8]

```
std::locale::locale (
 const locale & __base,
 const locale & __add,
 category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

#### Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                            |
| <code>__add</code>  | The locale to use facets from.                                 |
| <code>__cat</code>  | Set of categories defining the facets to use from <i>add</i> . |

References [locale\(\)](#).

#### locale() [8/8]

```
template<typename _Facet>
std::locale::locale (
 const locale & __other,
 _Facet * __f)
```

Construct locale with another facet.

Constructs a copy of the locale *\_\_other*. The facet *\_\_f* is added to *\_\_other*, replacing an existing facet of type *Facet* if there is one. If *\_\_f* is null, this locale is a copy of *\_\_other*.



**Parameters**

|                      |                      |
|----------------------|----------------------|
| <code>__other</code> | The locale to copy.  |
| <code>__f</code>     | The facet to add in. |

References [locale\(\)](#).

**~locale()**

```
std::locale::~~locale () throw ()
```

Locale destructor.

References [locale\(\)](#).

**6.657.4 Member Function Documentation****classic()**

```
const locale & std::locale::classic () [static], [nodiscard]
```

Return reference to the C locale.

**combine()**

```
template<typename _Facet>
locale std::locale::combine (
 const locale & __other) const [nodiscard]
```

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type `Facet` from the locale *other* into the new locale.

**Template Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| <code>_Facet</code> | The facet type to copy from other |
|---------------------|-----------------------------------|

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__other</code> | The locale to copy from. |
|----------------------|--------------------------|

**Returns**

Newly constructed locale.

**Exceptions**

|                                                 |                                                                    |
|-------------------------------------------------|--------------------------------------------------------------------|
| <a href="#"><code>std::runtime_error</code></a> | if <code>__other</code> has no facet of type <code>_Facet</code> . |
|-------------------------------------------------|--------------------------------------------------------------------|

References [locale\(\)](#).

Referenced by [operator=\(\)](#).

**global()**

```
locale std::locale::global (
 const locale & __loc) [static]
```

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__loc</code> | The new locale to make global. |
|--------------------|--------------------------------|

**Returns**

Copy of the old global locale.

References [locale\(\)](#).

**name()**

```
string std::locale::name () const [nodiscard]
```

Return locale name.

**Returns**

Locale name or "\*" if unnamed.

References [name\(\)](#).

Referenced by [name\(\)](#).

**operator()()**

```
template<typename _Char, typename _Traits, typename _Alloc>
bool std::locale::operator() (
 const basic_string< _Char, _Traits, _Alloc > & __s1,
 const basic_string< _Char, _Traits, _Alloc > & __s2) const [nodiscard]
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector `v` of strings could be sorted according to locale `loc` by doing:

```
std::sort(v.begin(), v.end(), loc);
```

**Parameters**

|                   |                           |
|-------------------|---------------------------|
| <code>__s1</code> | First string to compare.  |
| <code>__s2</code> | Second string to compare. |

**Returns**

True if `collate<_Char>` facet compares `__s1 < __s2`, else false.

References [locale\(\)](#).

**operator=()**

```
const locale & std::locale::operator= (
 const locale & __other) throw ()
```

Assignment operator.

Set this locale to be a copy of *other*.

**Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

**Returns**

A reference to this locale.

References [locale\(\)](#), and [combine\(\)](#).

**operator==()**

```
bool std::locale::operator== (
 const locale & __other) const throw () [nodiscard]
```

Locale equality.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

**Returns**

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

References [locale\(\)](#), and [operator==\(\)](#).

Referenced by [operator==\(\)](#).

**6.657.5 Friends And Related Symbol Documentation****has\_facet**

```
template<typename _Facet>
bool has_facet (
 const locale & __loc) throw () [friend]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Template Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

**Returns**

true if `__loc` contains a facet of type `_Facet`, else false.

References [locale\(\)](#).

**use\_facet**

```
template<typename _Facet>
const _Facet & use_facet (
 const locale & __loc) [friend]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_↵  
facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

**Template Parameters**

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

**Returns**

Reference to facet of type `Facet`.

**Exceptions**

|                               |                                                                             |
|-------------------------------|-----------------------------------------------------------------------------|
| <a href="#">std::bad_cast</a> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|-------------------------------|-----------------------------------------------------------------------------|

References [locale\(\)](#).

**6.657.6 Member Data Documentation****all**

```
const category std::locale::all [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

**collate**

```
const category std::locale::collate [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### ctype

```
const category std::locale::ctype [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### messages

```
const category std::locale::messages [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### monetary

```
const category std::locale::monetary [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### none

```
const category std::locale::none [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### numeric

```
const category std::locale::numeric [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

### time

```
const category std::locale::time [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 6.658 std::lock\_guard< \_Mutex > Class Template Reference

```
#include <mutex>
```

## Public Types

- typedef `_Mutex` **mutex\_type**

## Public Member Functions

- **lock\_guard** (const lock\_guard &)=delete
- **lock\_guard** (mutex\_type &\_\_m)
- **lock\_guard** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- lock\_guard & **operator=** (const lock\_guard &)=delete

### 6.658.1 Detailed Description

```
template<typename _Mutex>
class std::lock_guard<_Mutex >
```

A simple scoped lock type.

A lock\_guard controls mutex ownership within a scope, releasing ownership in the destructor.

Since

C++11

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 6.659 std::logic\_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::logic\_error:



## Public Member Functions

- `logic_error` (const char \*)
- `logic_error` (const `logic_error` &) noexcept
- `logic_error` (const `string` & \_\_arg)
- `logic_error` (`logic_error` &&) noexcept
- `logic_error` & `operator=` (const `logic_error` &) noexcept
- `logic_error` & `operator=` (`logic_error` &&) noexcept
- virtual const char \* `what` () const noexcept

### 6.659.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

### 6.659.2 Constructor & Destructor Documentation

#### `logic_error()`

```
std::logic_error::logic_error (
 const string & __arg) [explicit]
```

Takes a character string describing the error.

References `logic_error()`.

Referenced by `logic_error()`.

### 6.659.3 Member Function Documentation

#### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from `std::exception`.

Reimplemented in `std::future_error`.

References `what()`.

Referenced by `what()`.

The documentation for this class was generated from the following file:

- `stdexcept_except.h`

## 6.660 `std::logical_and<_Tp>` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::logical_and< _Tp >`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.660.1 Detailed Description

```
template<typename _Tp>
struct std::logical_and< _Tp >
```

One of the [Boolean operations functors](#).

#### 6.660.2 Member Typedef Documentation

##### **first\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### **result\_type**

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

##### **second\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



## 6.661 std::logical\_and< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_and< void >:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) && [std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) && [std::forward](#)< \_Up >(\_\_u))

#### 6.661.1 Detailed Description

One of the [Boolean operations functors](#).

#### 6.661.2 Member Typedef Documentation

##### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### result\_type

typedef bool [std::binary\\_function](#)< void, void, bool >::result\_type  
result\_type is the return type

##### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, bool >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.662 std::logical\_not< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_not< \_Tp >:



### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (const `_Tp` &\_\_x) const

#### 6.662.1 Detailed Description

```
template<typename _Tp>
struct std::logical_not< _Tp >
```

One of the [Boolean operations functors](#).

#### 6.662.2 Member Typedef Documentation

##### argument\_type

```
typedef _Tp std::unary_function< _Tp, bool >::argument_type [inherited]
argument_type is the type of the argument
```

##### result\_type

```
typedef bool std::unary_function< _Tp, bool >::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.663 `std::logical_not< void >` Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::logical_not< void >`:



### Public Types

- typedef void [argument\\_type](#)
- typedef `__is_transparent` `is_transparent`
- typedef bool [result\\_type](#)

### Public Member Functions

- template<typename `_Tp`>  
constexpr auto **operator()** (`_Tp` &&\_\_t) const noexcept(noexcept(![std::forward](#)< `_Tp` >(\_\_t))) -> decltype(![std::forward](#)< `_Tp` >(\_\_t))

#### 6.663.1 Detailed Description

One of the [Boolean operations functors](#).

#### 6.663.2 Member Typedef Documentation

##### **argument\_type**

typedef void [std::unary\\_function](#)< void, bool >::argument\_type  
argument\_type is the type of the argument

##### **result\_type**

typedef bool [std::unary\\_function](#)< void, bool >::result\_type  
result\_type is the return type

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.664 std::logical\_or< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_or< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.664.1 Detailed Description

```
template<typename _Tp>
struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

#### 6.664.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

##### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.665 std::logical\_or< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical\_or< void >:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)
- typedef bool [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t)||[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t)||[std::forward](#)< \_Up >(\_\_u))

#### 6.665.1 Detailed Description

One of the [Boolean operations functors](#).

#### 6.665.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

[first\\_argument\\_type](#) is the type of the first argument

##### result\_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

[result\\_type](#) is the return type

##### second\_argument\_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

[second\\_argument\\_type](#) is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.666 std::lognormal\_distribution< \_RealType > Class Template Reference

#include <random>

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

### Public Member Functions

- **lognormal\_distribution** (\_RealType \_\_m, \_RealType \_\_s=\_RealType(1))
- **lognormal\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- \_RealType **s** () const

### Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::lognormal\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [lognormal\\_distribution](#) &\_\_d1, const [lognormal\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::lognormal\\_distribution](#)< \_RealType1 > &\_\_x)

### 6.666.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::lognormal\_distribution< \_RealType >**

A lognormal\_distribution random number distribution.  
 The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Since

C++11

### 6.666.2 Member Typedef Documentation

#### result\_type

template<typename \_RealType = double>  
 typedef \_RealType **std::lognormal\_distribution< \_RealType >::result\_type**  
 The type of the range of the distribution.

### 6.666.3 Member Function Documentation

#### max()

template<typename \_RealType = double>  
**result\_type std::lognormal\_distribution< \_RealType >::max () const** [inline]  
 Returns the least upper bound value of the distribution.

#### min()

template<typename \_RealType = double>  
**result\_type std::lognormal\_distribution< \_RealType >::min () const** [inline]  
 Returns the greatest lower bound value of the distribution.

#### operator>()()

template<typename \_RealType = double>  
 template<typename \_UniformRandomNumberGenerator>  
**result\_type std::lognormal\_distribution< \_RealType >::operator() (**  
     \_uniformRandomNumberGenerator & \_\_urng) [inline]

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### param() [1/2]

template<typename \_RealType = double>  
**param\_type std::lognormal\_distribution< \_RealType >::param () const** [inline]  
 Returns the parameter set of the distribution.

#### param() [2/2]

template<typename \_RealType = double>  
 void **std::lognormal\_distribution< \_RealType >::param (**  
     const **param\_type** & \_\_param) [inline]

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### reset()

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::reset () [inline]
Resets the distribution state.
```

### 6.666.4 Friends And Related Symbol Documentation

#### operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::lognormal_distribution< _RealType1 > & __x) [friend]
Inserts a lognormal_distribution random number distribution __x into the output stream __os.
```

#### Parameters

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <code>__os</code> | An output stream.                                    |
| <code>__x</code>  | A lognormal_distribution random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### operator==

```
template<typename _RealType = double>
bool operator== (
 const lognormal_distribution< _RealType > & __d1,
 const lognormal_distribution< _RealType > & __d2) [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

#### operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::lognormal_distribution< _RealType1 > & __x) [friend]
Extracts a lognormal_distribution random number distribution __x from the input stream __is.
```



**Parameters**

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <code>__is</code> | An input stream.                                         |
| <code>__x</code>  | A lognormal_distribution random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.667 `__gnu_pbds::detail::lu_counter_metadata< Size_Type >` Class Template Reference**

```
#include <lu_counter_metadata.hpp>
```

**Public Types**

- typedef `Size_Type` **size\_type**

**Friends**

- class `lu_counter_policy_base< size_type >`

**6.667.1 Detailed Description**

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

**6.668 `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >` Class Template Reference**

```
#include <list_update_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`:



### Public Types

- enum { `max_count` }
- typedef `_Alloc allocator_type`
- typedef `detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference`
- typedef `detail::lu_counter_metadata< size_type > metadata_type`
- typedef `allocator_type::size_type size_type`

### Public Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_data) const`

### Private Member Functions

- `template<typename Metadata_Reference> bool operator() (Metadata_Reference r_data, size_type m_max_count) const`
- `lu_counter_metadata< size_type > operator() (size_type max_size) const`

#### 6.668.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

#### 6.668.2 Member Typedef Documentation

##### `metadata_reference`

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc, metadata_type>::reference __gnu_pbds::lu_counter_policy<
Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

**metadata\_type**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

**6.668.3 Member Enumeration Documentation****anonymous enum**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

**Enumerator**

|                        |                                                                                                |
|------------------------|------------------------------------------------------------------------------------------------|
| <code>max_count</code> | When some element is accessed this number of times, it will be moved to the front of the list. |
|------------------------|------------------------------------------------------------------------------------------------|

**6.668.4 Member Function Documentation****operator>() [1/2]**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const [inline]
```

Creates a metadata object.

References [max\\_count](#).

**operator>() [2/2]**

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (
 metadata_reference r_data) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

References [max\\_count](#).

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

**6.669 `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >` Class Template Reference**

```
#include <lu_counter_metadata.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >`:



### Protected Types

- typedef `Size_Type` **size\_type**

### Protected Member Functions

- template<typename Metadata\_Reference>  
bool **operator()** (Metadata\_Reference r\_data, size\_type m\_max\_count) const
- [lu\\_counter\\_metadata](#)< size\_type > **operator()** (size\_type max\_size) const

#### 6.669.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

#### 6.670 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

```
#include <lu_map_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**

- typedef point\_const\_iterator\_ **point\_const\_iterator**
- typedef point\_iterator\_ **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef stored\_data< value\_type, size\_type, Store\_Hash > **stored\_data\_type**
- typedef Update\_Policy::metadata\_type **update\_metadata**
- typedef Update\_Policy **update\_policy**
- typedef traits\_base::value\_type **value\_type**

### Public Member Functions

- **lu\_map** (const lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)
- template<typename It>  
  **lu\_map** (It, It)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
  size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference r\_key)
- point\_const\_iterator **find** (key\_const\_reference r\_key) const
- std::pair< point\_iterator, bool > **insert** (const\_reference)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy > &)

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Member Functions

- template<typename It>  
  void **copy\_from\_range** (It, It)

### Friends

- class **const\_iterator\_**
- class **iterator\_**

### 6.670.1 Detailed Description

```
template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >
```

list-based (with updates) associative container. Skip to the lu, my darling.  
 The documentation for this class was generated from the following file:

- [lu\\_map.hpp](#)

## 6.671 `__gnu_pbds::lu_move_to_front_policy<_Alloc>` Class Template Reference

```
#include <list_update_policy.hpp>
```

### Public Types

- typedef `_Alloc allocator_type`
- typedef `detail::rebind_traits<_Alloc, metadata_type>::reference metadata_reference`
- typedef `null_type metadata_type`

### Public Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference r_metadata) const`

### 6.671.1 Detailed Description

```
template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy<_Alloc>
```

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

### 6.671.2 Member Typedef Documentation

#### `metadata_reference`

```
template<typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc, metadata_type>::reference __gnu_pbds::lu_move_to_front_policy<
_Alloc>::metadata_reference
```

Reference to metadata on which this functor operates.

#### `metadata_type`

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_type
```

Metadata on which this functor operates.

### 6.671.3 Member Function Documentation

#### `operator()()` [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() () const [inline]
```

Creates a metadata object.

**operator>() [2/2]**

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() (
 metadata_reference r_metadata) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

**6.672 std::make\_signed< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

**6.672.1 Detailed Description**

```
template<typename _Tp>
struct std::make_signed< _Tp >
```

make\_signed

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**6.673 std::make\_unsigned< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using **type**

**6.673.1 Detailed Description**

```
template<typename _Tp>
struct std::make_unsigned< _Tp >
```

make\_unsigned

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

**6.674 \_\_gnu\_cxx::malloc\_allocator< \_Tp > Class Template Reference**

```
#include <malloc_allocator.h>
```

**Public Types**

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**



## Public Member Functions

- constexpr **malloc\_allocator** (const malloc\_allocator &) noexcept
- template<typename \_Tp1>  
constexpr **malloc\_allocator** (const malloc\_allocator< \_Tp1 > &) noexcept
- \_Tp \* **allocate** (size\_type \_\_n, const void \*=0)
- void **deallocate** (\_Tp \*\_\_p, size\_type)

## Friends

- template<typename \_Up>  
constexpr bool **operator==** (const malloc\_allocator &, const malloc\_allocator< \_Up > &) noexcept

### 6.674.1 Detailed Description

template<typename \_Tp>  
class **\_\_gnu\_cxx::malloc\_allocator**< \_Tp >

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

The documentation for this class was generated from the following files:

- [experimental/memory\\_resource](#)
- [malloc\\_allocator.h](#)

## 6.675 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference

#include <map.h>

Inheritance diagram for std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >:



## Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< \_Base\_const\_iterator, map > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< \_Base\_iterator, map > **iterator**

- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **map** (`_Base_ref __x`)
- template<typename `_InputIterator`>  
**map** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator`>  
**map** (`_InputIterator __first, _InputIterator __last, const allocator_type &__a`)
- **map** (`const _Compare &__comp, const _Allocator &__a=_Allocator()`)
- **map** (`const allocator_type &__a`)
- **map** (`const map &`)=default
- **map** (`const map &__m, const __type_identity_t< allocator_type > &__a`)
- **map** (`initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type()`)
- **map** (`initializer_list< value_type > __l, const allocator_type &__a`)
- **map** (`map &&`)=default
- **map** (`map &&__m, const __type_identity_t< allocator_type > &__a`) noexcept(noexcept(`_Base(std::move(__m), __a)`))
- `const _Base & _M_base` () const noexcept
- `_Base & _M_base` () noexcept
- `const_iterator begin` () const noexcept
- `iterator begin` () noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- template<typename... `_Args`>  
`std::pair< iterator, bool > emplace` (`_Args &&... __args`)
- template<typename... `_Args`>  
`iterator emplace_hint` (`const_iterator __pos, _Args &&... __args`)
- `const_iterator end` () const noexcept
- `iterator end` () noexcept
- `std::pair< iterator, iterator > equal_range` (`const key_type &__x`)
- `std::pair< const_iterator, const_iterator > equal_range` (`const key_type &__x`) const
- `_Base_iterator erase` (`_Base_const_iterator __position`)
- template<\_\_heterogeneous\_tree\_key< `map` > `_Kt`>  
`size_type erase` (`_Kt &&__x`)
- `size_type erase` (`const key_type &__x`)
- `iterator erase` (`const_iterator __first, const_iterator __last`)
- `iterator erase` (`const_iterator __position`)
- `iterator erase` (`iterator __position`)
- `iterator find` (`const key_type &__x`)

- const\_iterator **find** (const key\_type &\_\_x) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
[std::pair](#)< iterator, bool > **insert** (\_Pair &&\_\_x)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_list)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_x)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- map & **operator=** (const map &)=default
- map & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- map & **operator=** (map &&)=default
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- reverse\_iterator **rend** () noexcept
- void **swap** (map &\_\_x) noexcept(*/\*conditional \*/*)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::Safe\\_iterator](#)

#### 6.675.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator =
std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class std::map with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [map.h](#)

## 6.676 std::map< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

```
#include <map>
```

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- `map()`=default
- template<typename `_InputIterator`>  
`map` (`_InputIterator` `__first`, `_InputIterator` `__last`)
- template<typename `_InputIterator`>  
`map` (`_InputIterator` `__first`, `_InputIterator` `__last`, const `_Compare` &`__comp`, const `allocator_type` &`__a`↵  
`a=allocator_type()`)
- template<typename `_InputIterator`>  
`map` (`_InputIterator` `__first`, `_InputIterator` `__last`, const `allocator_type` &`__a`)
- `map` (const `_Compare` &`__comp`, const `allocator_type` &`__a`=`allocator_type()`)
- `map` (const `allocator_type` &`__a`)
- `map` (const `map` &)=default
- `map` (const `map` &`__m`, const `__type_identity_t`< `allocator_type` > &`__a`)
- `map` (`initializer_list`< `value_type` > `__l`, const `_Compare` &`__comp`=`_Compare()`, const `allocator_type` &`__a`↵  
`a=allocator_type()`)
- `map` (`initializer_list`< `value_type` > `__l`, const `allocator_type` &`__a`)
- `map` (`map` &&)=default
- `map` (`map` &&`__m`, const `__type_identity_t`< `allocator_type` > &`__a`) noexcept(`is_nothrow_copy_constructible`<  
`_Compare` >::value &&`_Alloc_traits::S_always_equal()`)
- `~map()`=default
- template<`__heterogeneous_tree_key`< `map` > `_Kt`>  
mapped\_type & `at` (const `_Kt` &`__k`)
- template<`__heterogeneous_tree_key`< `map` > `_Kt`>  
const mapped\_type & `at` (const `_Kt` &`__k`) const
- mapped\_type & `at` (const key\_type &`__k`)
- const mapped\_type & `at` (const key\_type &`__k`) const
- const\_iterator `begin` () const noexcept
- iterator `begin` () noexcept
- const\_iterator `cbegin` () const noexcept
- const\_iterator `cend` () const noexcept
- void `clear` () noexcept
- const\_reverse\_iterator `crbegin` () const noexcept
- const\_reverse\_iterator `crend` () const noexcept

- template<typename... \_Args>  
std::pair< iterator, bool > **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - const\_iterator **end** () const noexcept
  - iterator **end** () noexcept
  - iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type **get\_allocator** () const noexcept
  - template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** (std::initializer\_list< value\_type > \_\_list)
  - key\_compare **key\_comp** () const
  - size\_type **max\_size** () const noexcept
  - map & **operator=** (const map &)=default
  - map & **operator=** (initializer\_list< value\_type > \_\_l)
  - map & **operator=** (map &&)=default
  - template<\_\_heterogeneous\_tree\_key< map > \_Kt>  
mapped\_type & **operator[]** (\_Kt &&\_\_k)
  - mapped\_type & **operator[]** (const key\_type &\_\_k)
  - mapped\_type & **operator[]** (key\_type &&\_\_k)
  - const\_reverse\_iterator **rbegin** () const noexcept
  - reverse\_iterator **rbegin** () noexcept
  - const\_reverse\_iterator **rend** () const noexcept
  - reverse\_iterator **rend** () noexcept
  - size\_type **size** () const noexcept
  - void **swap** (map &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare **value\_comp** () const
- 
- std::pair< iterator, bool > **insert** (const value\_type &\_\_x)
  - std::pair< iterator, bool > **insert** (value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< **is\_constructible**< value\_type, \_Pair >::value, pair< iterator, bool > > **insert** (\_Pair &&\_\_x)
- 
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< **is\_constructible**< value\_type, \_Pair >::value, iterator > **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- 
- iterator **erase** (const\_iterator \_\_position)
  - iterator **erase** (iterator \_\_position)
- 
- size\_type **erase** (const key\_type &\_\_x)
  - template<\_\_heterogeneous\_tree\_key< map > \_Kt>  
size\_type **erase** (\_Kt &&\_\_x)

- iterator `find` (const key\_type &\_\_x)
- const\_iterator `find` (const key\_type &\_\_x) const
- size\_type `count` (const key\_type &\_\_x) const
- bool `contains` (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto `contains` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- iterator `lower_bound` (const key\_type &\_\_x)
- const\_iterator `lower_bound` (const key\_type &\_\_x) const
- iterator `upper_bound` (const key\_type &\_\_x)
- const\_iterator `upper_bound` (const key\_type &\_\_x) const
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1>  
\_\_detail::\_\_synth3way\_t< `pair`< const \_K1, \_T1 > > `operator`<=> (const `map`< \_K1, \_T1, \_C1, \_A1 > &, const `map`< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1>  
bool `operator`== (const `map`< \_K1, \_T1, \_C1, \_A1 > &, const `map`< \_K1, \_T1, \_C1, \_A1 > &)

### 6.676.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::↵
allocator<std::pair<const _Key, _Tp> >>
class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

### Template Parameters

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                |
| <code>_Tp</code>      | Type of mapped objects.                                                             |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .        |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 6.676.2 Constructor & Destructor Documentation

#### `map()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::map<_Key, _Tp, _Compare, _Alloc >::map () [default]
```

Default constructor creates no elements.

#### `map()` [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a map with no elements.

### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

#### `map()` [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 const map<_Key, _Tp, _Compare, _Alloc > &) [default]
```

Map copy constructor.

Whether the allocator is copied depends on the allocator traits.

**map()** [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 map< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Map move constructor.

The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

**map()** [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a map from an `initializer_list`.

**Parameters**

|                     |                                    |
|---------------------|------------------------------------|
| <code>__l</code>    | An <code>initializer_list</code> . |
| <code>__comp</code> | A comparison object.               |
| <code>__a</code>    | An allocator object.               |

Create a map consisting of copies of the elements in the `initializer_list __l`. This is linear in  $N$  if the range is already sorted, and  $N \log N$  otherwise (where  $N$  is `__l.size()`).

**map()** [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

**map()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const map< _Key, _Tp, _Compare, _Alloc > & __m,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

**map()** [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
```



```
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 map<_Key, _Tp, _Compare, _Alloc > && __m,
 const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

#### map() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

#### map() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

#### map() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a map from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in *N* if the range is already sorted, and *N*log*N* otherwise (where *N* is `distance(__first, __last)`).

#### map() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map<_Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a map from a range.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

#### `~map()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::map< _Key, _Tp, _Compare, _Alloc >::~~map () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### 6.676.3 Member Function Documentation

#### `at()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::at (
 const key_type & __k) [inline]
```

Access to map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

If a heterogeneous key `__k` matches a range of elements, the first is chosen.

#### `begin()` [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

### begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

### cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

### cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

### clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map<_Key, _Tp, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map<_Key, _Tp, _Compare, _Alloc >::contains (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

### Returns

True if there is an element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map<_Key, _Tp, _Compare, _Alloc >::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

True if there is an element with the specified key.

**count()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map<_Key, _Tp, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

**crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the std::pair built from `__args` (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Referenced by `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace()`, and `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`.

**empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map<_Key, _Tp, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the map is empty. (Thus begin() would equal end().)

**end() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

**end() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Referenced by `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at()`, `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`, and `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`.

**equal\_range() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::map<_Key, _Tp, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

**equal\_range() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::map<_Key, _Tp, _Compare, _Alloc >::equal_range
(
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↔</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

#### erase() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<__heterogeneous_tree_key< map > _Kt>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 _Kt && __x) [inline]
```

Erases elements according to the provided key.

#### Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | Key of element to be erased. |
| <code>_X</code> |                              |

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### erase() [2/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

#### Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | Key of element to be erased. |
| <code>_X</code> |                              |

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [3/5]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a map.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [4/5]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [5/5]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```



```
iterator std::map<_Key, _Tp, _Compare, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from a map.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### find() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a map.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

#### find() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a map.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

**insert() [1/8]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert() [2/8]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > std::map<
_Key, _Tp, _Compare, _Alloc >::insert (
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Referenced by `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp >>>::insert()`, and `std::map<_Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp >>>::operator[]()`.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::map< _Key, _Tp, _C
Compare, _Alloc >::insert (
 const_iterator __position,
 _Pair && __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
void std::map<_Key, _Tp, _Compare, _Alloc>::insert (
 std::initializer_list<value_type> & __list) [inline]
```

Attempts to insert a list of std::pairs into the map.

**Parameters**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__list</code> | A std::initializer_list<value_type> of pairs to be inserted. |
|---------------------|--------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>::insert (
 value_type && __x) [inline]
```

Attempts to insert a std::pair into the map.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. Insertion requires logarithmic time.

**key\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
key_compare std::map<_Key, _Tp, _Compare, _Alloc>::key_comp () const [inline]
```

Returns the key comparison object out of which the map was constructed.

Referenced by [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator<pair<const \\_Key, \\_Tp>>>::at\(\)](#), [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator<pair<const \\_Key, \\_Tp>>>::insert\(\)](#), and [std::map<\\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator<pair<const \\_Key, \\_Tp>>>::operator\[\]](#).

**lower\_bound()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↔</code> | Key of (key, value) pair to be located. |
| <code>_x</code> |                                         |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Referenced by `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at()`, `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert()`, and `std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::lower_bound()`.

**lower\_bound()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↔</code> | Key of (key, value) pair to be located. |
| <code>_x</code> |                                         |

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the map.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
 const map< _Key, _Tp, _Compare, _Alloc > &) [default]
```

Map assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Map list assignment operator.

#### Parameters

|   |                      |
|---|----------------------|
| ↔ | An initializer_list. |
| ↔ |                      |
| ↔ |                      |
| ↔ |                      |
| / |                      |

This function fills a map with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

#### operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
 map< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

#### operator[]()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (
 const key_type & __k) [inline]
```

Subscript ( [] ) access to map data.

#### Parameters

|    |                                             |
|----|---------------------------------------------|
| ↔  | The key for which data should be retrieved. |
| _k |                                             |

#### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

If a heterogeneous key matches a range of elements, the first is chosen.

Lookup requires logarithmic time.

#### rbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

**rend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**rend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

**size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the map.

**swap()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
 map< _Key, _Tp, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another map.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <a href="#">_X</a> | A map of the same element and allocator types. |
|--------------------|------------------------------------------------|

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

**upper\_bound()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```



Finds the end of a subsequence matching given key.

#### Parameters

|                                 |                                         |
|---------------------------------|-----------------------------------------|
| <a href="#"><code>_↔</code></a> | Key of (key, value) pair to be located. |
| <a href="#"><code>_X</code></a> |                                         |

#### Returns

Iterator pointing to the first element greater than key, or end().

#### upper\_bound() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                                 |                                         |
|---------------------------------|-----------------------------------------|
| <a href="#"><code>_↔</code></a> | Key of (key, value) pair to be located. |
| <a href="#"><code>_X</code></a> |                                         |

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

#### value\_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map<_Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

## 6.677 std::mask\_array<\_Tp> Class Template Reference

```
#include <mask_array.h>
```

#### Public Types

- typedef `_Tp` **value\_type**

#### Public Member Functions

- [`mask\_array`](#) (const `mask_array` &)
- template<class `_Dom`>  
void **operator%=>** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [`operator%=>`](#) (const [`valarray`](#)< `_Tp` > &) const

- `template<class _Dom>`  
`void operator&= (const _Expr< _Dom, _Tp > &) const`
- `void operator&= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator*= (const _Expr< _Dom, _Tp > &) const`
- `void operator*= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Ex>`  
`void operator= (const _Expr< _Ex, _Tp > &__e) const`
- `void operator= (const _Tp &) const`
- `mask_array & operator= (const mask_array &)`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `template<class _Dom>`  
`void operator|= (const _Expr< _Dom, _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`

## Friends

- `class valarray< _Tp >`

### 6.677.1 Detailed Description

**template<class \_Tp>**

**class std::mask\_array< \_Tp >**

Reference to selected subset of an array.

A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of `bool`. The way to get a `mask_array` is to call `operator[](valarray<bool>)` on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array `(false, true, false, true)` as an argument, the mask array has two elements referring to `array[1]` and `array[3]` in the underlying array.

## Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

### 6.677.2 Member Function Documentation

#### operator%=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator%= (
 const valarray<_Tp> &) const
```

Modulo slice elements by corresponding elements of *v*.

#### operator&=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator&= (
 const valarray<_Tp> &) const
```

Logical and slice elements with corresponding elements of *v*.

#### operator\*=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator*= (
 const valarray<_Tp> &) const
```

Multiply slice elements by corresponding elements of *v*.

#### operator+=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator+= (
 const valarray<_Tp> &) const
```

Add corresponding elements of *v* to slice elements.

#### operator-=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator-= (
 const valarray<_Tp> &) const
```

Subtract corresponding elements of *v* from slice elements.

#### operator/=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator/= (
 const valarray<_Tp> &) const
```

Divide slice elements by corresponding elements of *v*.

#### operator<<=( )

```
template<class _Tp>
void std::mask_array<_Tp>::operator<<= (
 const valarray<_Tp> &) const
```

Left shift slice elements by corresponding elements of *v*.

**operator>>=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator>>= (
 const valarray< _Tp > &) const
```

Right shift slice elements by corresponding elements of *v*.

**operator^=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator^= (
 const valarray< _Tp > &) const
```

Logical xor slice elements with corresponding elements of *v*.

**operator" |=()**

```
template<class _Tp>
void std::mask_array< _Tp >::operator|= (
 const valarray< _Tp > &) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [mask\\_array.h](#)

## 6.678 \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type > Class Template Reference

```
#include <mask_based_range_hashing.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type >:

**Protected Types**

- typedef Size\_Type **size\_type**

**Protected Member Functions**

- void **notify\_resized** (size\_type size)
- size\_type **range\_hash** (size\_type hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

**6.678.1 Detailed Description**

```
template<typename Size_Type>
```

```
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

The documentation for this class was generated from the following file:

- [mask\\_based\\_range\\_hashing.hpp](#)

**6.679 std::match\_results< \_Bi\_iter, \_Alloc > Class Template Reference**

```
#include <regex>
```

Inheritance diagram for std::match\_results< \_Bi\_iter, \_Alloc >:

**Public Types****28.10 Public Types**

- typedef [sub\\_match](#)< \_Bi\_iter > **value\_type**
- typedef const value\_type & **const\_reference**
- typedef value\_type & **reference**
- typedef \_Base\_type::const\_iterator **const\_iterator**
- typedef const\_iterator **iterator**
- typedef \_\_iter\_traits::difference\_type **difference\_type**
- typedef [allocator\\_traits](#)< \_Alloc >::size\_type **size\_type**
- typedef \_Alloc **allocator\_type**

- typedef \_\_iter\_traits::value\_type **char\_type**
- typedef [std::basic\\_string](#)< char\_type > **string\_type**

## Public Member Functions

- template<typename \_Out\_iter>  
\_Out\_iter **format** (\_Out\_iter \_\_out, const [match\\_results](#)< \_Bi\_iter, \_Alloc >::char\_type \*\_\_fmt\_first, const [match\\_results](#)< \_Bi\_iter, \_Alloc >::char\_type \*\_\_fmt\_last, [match\\_flag\\_type](#) \_\_flags) const
- bool [ready](#) () const noexcept

### 28.10.1 Construction, Copying, and Destruction

- [match\\_results](#) ()
- [match\\_results](#) (const \_Alloc &\_\_a) noexcept
- [match\\_results](#) (const [match\\_results](#) &)=default
- [match\\_results](#) ([match\\_results](#) &&) noexcept=default
- [match\\_results](#) & [operator=](#) (const [match\\_results](#) &)=default
- [match\\_results](#) & [operator=](#) ([match\\_results](#) &&)=default
- ~[match\\_results](#) ()=default
- [match\\_results](#) (const [match\\_results](#) &\_\_m, const \_Alloc &\_\_a)
- [match\\_results](#) ([match\\_results](#) &&\_\_m, const \_Alloc &\_\_a) noexcept(noexcept(\_Base\_type([std::move](#)(\_\_m), ←  
\_\_a)))

### 28.10.2 Size

- size\_type [size](#) () const noexcept
- size\_type [max\\_size](#) () const noexcept
- bool [empty](#) () const noexcept

### 28.10.4 Element Access

- difference\_type [length](#) (size\_type \_\_sub=0) const
- difference\_type [position](#) (size\_type \_\_sub=0) const
- string\_type [str](#) (size\_type \_\_sub=0) const
- const\_reference [operator\[\]](#) (size\_type \_\_sub) const
- const\_reference [prefix](#) () const
- const\_reference [suffix](#) () const
- const\_iterator [begin](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [end](#) () const noexcept
- const\_iterator [cend](#) () const noexcept

### 28.10.5 Formatting

*These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.*

- template<typename \_Out\_iter>  
\_Out\_iter [format](#) (\_Out\_iter \_\_out, const char\_type \*\_\_fmt\_first, const char\_type \*\_\_fmt\_last, [match\\_flag\\_type](#) \_\_flags=[regex\\_constants::format\\_default](#)) const
- template<typename \_Out\_iter, typename \_St, typename \_Sa>  
\_Out\_iter [format](#) (\_Out\_iter \_\_out, const [basic\\_string](#)< char\_type, \_St, \_Sa > &\_\_fmt, [match\\_flag\\_type](#) \_\_←  
flags=[regex\\_constants::format\\_default](#)) const
- template<typename \_St, typename \_Sa>  
[basic\\_string](#)< char\_type, \_St, \_Sa > [format](#) (const [basic\\_string](#)< char\_type, \_St, \_Sa > &\_\_fmt, [match\\_flag\\_type](#) \_\_flags=[regex\\_constants::format\\_default](#)) const
- string\_type [format](#) (const char\_type \*\_\_fmt, [match\\_flag\\_type](#) \_\_flags=[regex\\_constants::format\\_default](#)) const

### 28.10.6 Allocator

- allocator\_type [get\\_allocator](#) () const noexcept

### 28.10.7 Swap

- void [swap](#) (match\_results &\_\_that) noexcept

### Private Member Functions

- constexpr iterator [begin](#) () noexcept
- constexpr iterator [end](#) () noexcept
- constexpr const\_reference [operator\[\]](#) (size\_type \_\_n) const noexcept
- constexpr reference [operator\[\]](#) (size\_type \_\_n) noexcept
- constexpr void [swap](#) (vector &\_\_x) noexcept

### Friends

- template<typename, typename, typename>  
class **regex\_iterator**

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_Bi\_iter, typename \_Alloc>  
bool [operator==](#) (const [match\\_results](#)<\_Bi\_iter, \_Alloc > &\_\_m1, const [match\\_results](#)<\_Bi\_iter, \_Alloc > &\_\_m2)
- template<typename \_Bi\_iter, typename \_Alloc>  
void [swap](#) ([match\\_results](#)<\_Bi\_iter, \_Alloc > &\_\_lhs, [match\\_results](#)<\_Bi\_iter, \_Alloc > &\_\_rhs) noexcept

### 6.679.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results<_Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template match\_results.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The sub\_match object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the sub\_match member matched is always true. The sub\_match object stored at index n denotes what matched the marked sub-expression n within the matched expression. If the sub-expression n participated in a regular expression match then the sub\_match member matched evaluates to true, and members first and second denote the range of characters [first, second) which formed that match. Otherwise matched is false, and members first and second point to the end of the sequence that was searched.

Since

C++11

## 6.679.2 Constructor & Destructor Documentation

### match\_results() [1/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results () [inline]
```

Constructs a default match\_results container.

#### Postcondition

size() returns 0 and str() returns an empty string.

### match\_results() [2/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 const _Alloc & __a) [inline], [explicit], [noexcept]
```

Constructs a default match\_results container.

#### Postcondition

size() returns 0 and str() returns an empty string.

### match\_results() [3/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 const match_results< _Bi_iter, _Alloc > &) [default]
```

Copy constructs a match\_results.

### match\_results() [4/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 match_results< _Bi_iter, _Alloc > &&) [default], [noexcept]
```

Move constructs a match\_results.

### ~match\_results()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::~~match_results () [default]
```

Destroys a match\_results object.

### match\_results() [5/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 const match_results< _Bi_iter, _Alloc > & __m,
 const _Alloc & __a) [inline]
```

Constructs a default match\_results container.

#### Postcondition

size() returns 0 and str() returns an empty string.



**match\_results()** [6/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
std::match_results<_Bi_iter, _Alloc>::match_results (
 match_results<_Bi_iter, _Alloc> && __m,
 const _Alloc & __a) [inline], [noexcept]
```

Constructs a default match\_results container.

**Postcondition**

size() returns 0 and str() returns an empty string.

**6.679.3 Member Function Documentation****begin()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_iterator std::match_results<_Bi_iter, _Alloc>::begin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

Referenced by [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>>::const char\\*>::cbegin\(\)](#), and [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>>::const char\\*>::operator==\(\)](#).

**cbegin()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_iterator std::match_results<_Bi_iter, _Alloc>::cbegin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

**cend()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_iterator std::match_results<_Bi_iter, _Alloc>::cend () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

**empty()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
bool std::match_results<_Bi_iter, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Indicates if the match\_results contains no results.

**Return values**

|              |                                        |
|--------------|----------------------------------------|
| <i>true</i>  | The match_results object is empty.     |
| <i>false</i> | The match_results object is not empty. |

Referenced by [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>>::const char\\*>::operator==\(\)](#), [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>>::const char\\*>::prefix\(\)](#), and [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>>::const char\\*>::suffix\(\)](#).

**end()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_iterator std::match_results<_Bi_iter, _Alloc>::end () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator>>>>::const char*>::cend()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator>>>>::const char*>::operator==()`.

#### **format() [1/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
template<typename _Out_iter, typename _St, typename _Sa>
_Out_iter std::match_results<_Bi_iter, _Alloc>::format (
 _Out_iter __out,
 const basic_string< char_type, _St, _Sa > & __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

#### **Precondition**

`ready() == true`

#### **format() [2/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
template<typename _Out_iter>
_Out_iter std::match_results<_Bi_iter, _Alloc>::format (
 _Out_iter __out,
 const char_type * __fmt_first,
 const char_type * __fmt_last,
 match_flag_type __flags = regex_constants::format_default) const
```

#### **Precondition**

`ready() == true`

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator>>>>::const char*>::format()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator>>>>::const char*>::format()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator< sub_match<_BidirectionalIterator>>>>::const char*>::format()`.

#### **format() [3/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
template<typename _St, typename _Sa>
basic_string< char_type, _St, _Sa > std::match_results<_Bi_iter, _Alloc>::format (
 const basic_string< char_type, _St, _Sa > & __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

#### **Precondition**

`ready() == true`

#### **format() [4/4]**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
string_type std::match_results<_Bi_iter, _Alloc>::format (
 const char_type * __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

#### **Precondition**

`ready() == true`

**get\_allocator()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
allocator_type std::match_results<_Bi_iter, _Alloc>::get_allocator () const [inline], [noexcept]
```

Gets a copy of the allocator.

**length()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
difference_type std::match_results<_Bi_iter, _Alloc>::length (
 size_type __sub = 0) const [inline]
```

Gets the length of the indicated submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

`ready() == true`

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

**max\_size()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
size_type std::match_results<_Bi_iter, _Alloc>::max_size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

**operator=()** [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
match_results & std::match_results<_Bi_iter, _Alloc>::operator= (
 const match_results<_Bi_iter, _Alloc> &) [default]
```

Assigns rhs to \*this.

**operator=()** [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
match_results & std::match_results<_Bi_iter, _Alloc>::operator= (
 match_results<_Bi_iter, _Alloc> &&) [default]
```

Move-assigns rhs to \*this.

**operator[]()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_reference std::match_results<_Bi_iter, _Alloc>::operator[] (
 size_type __sub) const [inline]
```

Gets a `sub_match` reference for the match or submatch.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

#### Precondition

`ready() == true`

This function gets a reference to the indicated submatch, or the entire match if `__sub` is zero.  
If `__sub >= size()` then this function returns a `sub_match` with a special value indicating no submatch.

#### `position()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
 size_type __sub = 0) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

#### Precondition

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

#### `prefix()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::prefix () const [inline]
```

Gets a `sub_match` representing the match prefix.

#### Precondition

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Referenced by [std::match\\_results< \\_BidirectionalIterator, polymorphic\\_allocator< sub\\_match< \\_BidirectionalIterator > >>::operator==\( \)](#).

#### `ready()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::ready () const [inline], [noexcept]
```

Indicates if the `match_results` is ready.

**Return values**

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>true</i>  | The object has a fully-established result state. |
| <i>false</i> | The object is not ready.                         |

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator==(, std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator[ ]()`, `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::prefix()`, and `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::suffix()`.

**size()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
size_type std::match_results<_Bi_iter, _Alloc>::size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count()` + 1 if a match was successful. Some matches may be empty.

**Returns**

the number of matches found.

Referenced by `std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator==(, and std::match_results<_BidirectionalIterator, polymorphic_allocator<sub_match<_BidirectionalIterator>>><const char*>::operator[ ]().`

**str()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
string_type std::match_results<_Bi_iter, _Alloc>::str (
 size_type __sub = 0) const [inline]
```

Gets the match or submatch converted to a string type.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

`ready() == true`

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

**suffix()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
const_reference std::match_results<_Bi_iter, _Alloc>::suffix () const [inline]
```

Gets a `sub_match` representing the match suffix.

**Precondition**

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Referenced by [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>::operator==\(\)](#).

**swap()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
void std::match_results<_Bi_iter, _Alloc>::swap (
 match_results<_Bi_iter, _Alloc> & __that) [inline], [noexcept]
```

Swaps the contents of two `match_results`.

Referenced by [std::match\\_results<\\_BidirectionalIterator, polymorphic\\_allocator<sub\\_match<\\_BidirectionalIterator>>>::operator==\(\)](#).

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

## 6.680 `__gnu_pbds::detail::maybe_null_type<Key, Mapped, _Alloc, Store_Hash>` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >`:



### 6.680.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
```

Base class for conditionally defining a static data member.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.681 `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >`:



### Static Public Attributes

- static `null_type` `s_null_type`

#### 6.681.1 Detailed Description

```
template<typename Key, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
```

Specialization that defines a static data member of type `null_type`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.682 `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

```
#include <stl_function.h>
```



Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

### Public Member Functions

- `mem_fun1_ref_t` (`_Ret` (`_Tp::*`\_\_pf) (`_Arg`))
- `_Ret operator()` (`_Tp &`\_\_r, `_Arg` \_\_x) const

#### 6.682.1 Detailed Description

template<typename `_Ret`, typename `_Tp`, typename `_Arg`>  
 class std::mem\_fun1\_ref\_t< `_Ret`, `_Tp`, `_Arg` >

One of the [adaptors for member pointers](#).

#### 6.682.2 Member Typedef Documentation

##### first\_argument\_type

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::first\_argument\_type [inherited]  
 first\_argument\_type is the type of the first argument

##### result\_type

typedef `_Ret` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::result\_type [inherited]  
 result\_type is the return type

##### second\_argument\_type

typedef `_Arg` [std::binary\\_function](#)< `_Tp`, `_Arg`, `_Ret` >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 6.683 std::mem\_fun1\_t<\_Ret, \_Tp, \_Arg> Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun1\_t<\_Ret, \_Tp, \_Arg>:



### Public Types

- typedef \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

### Public Member Functions

- **mem\_fun1\_t** (\_Ret(\_Tp::\* \_\_pf)(\_Arg))
- **operator()** (\_Tp \* \_\_p, \_Arg \_\_x) const

#### 6.683.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

#### 6.683.2 Member Typedef Documentation

##### first\_argument\_type

```
typedef _Tp * std::binary_function<_Tp *, _Arg, _Ret>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### result\_type

```
typedef _Ret std::binary_function<_Tp *, _Arg, _Ret>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Arg std::binary_function< _Tp *, _Arg, _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**6.684 std::mem\_fun\_ref\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::mem\_fun\_ref\_t< \_Ret, \_Tp >:

**Public Types**

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

**Public Member Functions**

- `mem_fun_ref_t` (`_Ret` (`_Tp::*__pf`)())
- `_Ret operator()` (`_Tp &__r`) const

**6.684.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**6.684.2 Member Typedef Documentation****argument\_type**

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument

**result\_type**

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

**6.685 std::mem\_fun\_t< \_Ret, \_Tp > Class Template Reference**

```
#include <std_function.h>
```

Inheritance diagram for `std::mem_fun_t< _Ret, _Tp >`:

**Public Types**

- typedef `_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

**Public Member Functions**

- `mem_fun_t` (`_Ret` (`_Tp::*` `__pf`)())
- `_Ret operator()` (`_Tp *` `__p`) const

**6.685.1 Detailed Description**

```
template<typename _Ret, typename _Tp>
class std::mem_fun_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

**6.685.2 Member Typedef Documentation****argument\_type**

```
typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef _Ret std::unary_function< _Tp *, _Ret >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

**6.686 std::pmr::memory\_resource Class Reference**

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::memory\_resource:

**Public Member Functions**

- **memory\_resource** (const memory\_resource &)=default
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \*\_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const memory\_resource &\_\_other) const noexcept
- memory\_resource & **operator=** (const memory\_resource &)=default

**6.686.1 Detailed Description**

Class memory\_resource.

Since

C++17

The documentation for this class was generated from the following file:

- [memory\\_resource.h](#)

**6.687 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f > Class Template Reference**

```
#include <random>
```

**Public Types**

- typedef \_UIntType [result\\_type](#)

## Public Member Functions

- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`mersenne_twister_engine` (`_Sseq &__q`)
- `mersenne_twister_engine` (`result_type __sd`)
- `void discard` (`unsigned long long __z`)
- `result_type operator()` ()
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq>` `seed` (`_Sseq &__q`)
- `void seed` (`result_type __sd=default_seed`)

## Static Public Member Functions

- `static constexpr result_type max` ()
- `static constexpr result_type min` ()

## Static Public Attributes

- `static constexpr result_type default_seed`
- `static constexpr result_type initialization_multiplier`
- `static constexpr size_t mask_bits`
- `static constexpr size_t shift_size`
- `static constexpr size_t state_size`
- `static constexpr result_type tempering_b`
- `static constexpr result_type tempering_c`
- `static constexpr result_type tempering_d`
- `static constexpr size_t tempering_l`
- `static constexpr size_t tempering_s`
- `static constexpr size_t tempering_t`
- `static constexpr size_t tempering_u`
- `static constexpr size_t word_size`
- `static constexpr result_type xor_mask`

## Friends

- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits>` `& operator<<` (`std::basic_ostream<_CharT, _Traits> &__os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x`)
- `bool operator==` (`const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs`)
- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits>` `& operator>>` (`std::basic_istream<_CharT, _Traits> &__is, std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> &__x`)

### 6.687.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>
{
public:
 typedef _UIntType result_type;
 static constexpr result_type default_seed = 0;
 static constexpr result_type initialization_multiplier = 6364136223846793005ULL;
 static constexpr size_t mask_bits = 31;
 static constexpr size_t shift_size = 1;
 static constexpr size_t state_size = 32;
 static constexpr result_type tempering_b = 0x55555555;
 static constexpr result_type tempering_c = 0x55555555;
 static constexpr result_type tempering_d = 0x55555555;
 static constexpr size_t tempering_l = 1;
 static constexpr size_t tempering_s = 1;
 static constexpr size_t tempering_t = 1;
 static constexpr size_t tempering_u = 1;
 static constexpr size_t word_size = 32;
 static constexpr result_type xor_mask = 0x55555555;
 mersenne_twister_engine() {}
 mersenne_twister_engine(result_type __sd) {}
 mersenne_twister_engine(const mersenne_twister_engine &__q) {}
 mersenne_twister_engine& operator=(const mersenne_twister_engine &__q) {}
 void discard(unsigned long long __z) {}
 result_type operator()() {}
 void seed(result_type __sd=default_seed) {}
 void seed(const mersenne_twister_engine &__q) {}
 static constexpr result_type max() {}
 static constexpr result_type min() {}
};
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined `mt19937` class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

### Template Parameters

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__w</code> | Word size, the number of bits in each element of the state vector. |
| <code>__n</code> | The degree of recursion.                                           |
| <code>__m</code> | The period parameter.                                              |
| <code>__r</code> | The separation point bit index.                                    |
| <code>__a</code> | The last row of the twist matrix.                                  |
| <code>__u</code> | The first right-shift tempering matrix parameter.                  |
| <code>__d</code> | The first right-shift tempering matrix mask.                       |
| <code>__s</code> | The first left-shift tempering matrix parameter.                   |
| <code>__b</code> | The first left-shift tempering matrix mask.                        |
| <code>__t</code> | The second left-shift tempering matrix parameter.                  |
| <code>__c</code> | The second left-shift tempering matrix mask.                       |
| <code>__l</code> | The second right-shift tempering matrix parameter.                 |
| <code>__f</code> | Initialization multiplier.                                         |

Since

C++11

## 6.687.2 Member Typedef Documentation

### result\_type

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type
```

The type of the generated random value.

## 6.687.3 Constructor & Destructor Documentation

### mersenne\_twister\_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine (
 _Sseq & __q) [inline], [explicit]
```

Constructs a `mersenne_twister_engine` random number generator engine seeded from the seed sequence `__q`.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

## 6.687.4 Member Function Documentation

### discard()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::discard (
 unsigned long long __z)
```

Discard a sequence of random numbers.

### max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::max () [inline], [static], [constexpr]
```

Gets the largest possible value in the output range.



## min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```
constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min () [inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.

## 6.687.5 Friends And Related Symbol Documentation

### operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]
```

Inserts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x into the output stream \_\_os.

### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__os</code> | An output stream.                                           |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

### Returns

The output stream with the state of \_\_x inserted or in an error state.

### operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```
bool operator== (
 const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs,
 const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs) [friend]
```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

### Parameters

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| <code>__lhs</code> | A % mersenne_twister_engine random number generator object.       |
| <code>__rhs</code> | Another % mersenne_twister_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]
```

Extracts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__is</code> | An input stream.                                            |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

**Returns**

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.688 std::messages<\_CharT > Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages<\_CharT>:



### Public Types

- typedef int **catalog**
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `messages` (`__c_locale __cloc`, `const char *__s`, `size_t __refs=0`)
- `messages` (`size_t __refs=0`)
- void **close** (`catalog __c`) const
- `string_type` **get** (`catalog __c`, `int __set`, `int __msgid`, `const string_type &__s`) const
- `catalog` **open** (`const basic_string<char> &`, `const locale &`, `const char *`) const
- `catalog` **open** (`const basic_string<char> &__s`, `const locale &__loc`) const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- virtual `~messages` ()
- `string_type` **M\_convert\_from\_char** (`char *`) const
- `char *` **M\_convert\_to\_char** (`const string_type &__msg`) const
- virtual void **do\_close** (`catalog`) const
- void **do\_close** (`catalog`) const
- void **do\_close** (`catalog`) const
- `string` **do\_get** (`catalog`, `int`, `int`, `const string &`) const
- virtual `string_type` **do\_get** (`catalog`, `int`, `int`, `const string_type &__default`) const

- `wstring do_get` (catalog, int, int, const `wstring` &) const
- virtual catalog `do_open` (const `basic_string`< char > &, const locale &) const
- `messages`< wchar\_t >::catalog `do_open` (const `basic_string`< char > &, const locale &) const
- `messages`< char >::catalog `do_open` (const `basic_string`< char > &, const locale &) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale` & \_\_cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` & \_\_cloc, const char \* \_\_s, `__c_locale` \_\_old=0)
- static void `_S_destroy_c_locale` (`__c_locale` & \_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale` \_\_cloc, const char \* \_\_s)

### Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char \* `_M_name_messages`

#### 6.688.1 Detailed Description

```
template<typename _CharT>
class std::messages< _CharT >
```

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar\_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

#### 6.688.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT>
typedef _CharT std::messages< _CharT >::char_type
```

Public typedefs.

##### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::messages< _CharT >::string_type
```

Public typedefs.

#### 6.688.3 Constructor & Destructor Documentation

##### messages() [1/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### messages() [2/2]

```
template<typename _CharT>
std::messages<_CharT>::messages (
 __c_locale __cloc,
 const char * __s,
 size_t __refs = 0) [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

#### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__s</code>    | The name of a locale.               |
| <code>__refs</code> | Refcount to pass to the base class. |

References [std::locale::facet::facet\(\)](#).

#### ~messages()

```
template<typename _CharT>
std::messages<_CharT>::~~messages () [protected], [virtual]
Destructor.
```

### 6.688.4 Member Function Documentation

#### do\_get()

```
string std::messages<char>::do_get (
 catalog ,
 int ,
 int ,
 const string &) const [protected]
```

Specializations for required instantiations.

### 6.688.5 Member Data Documentation

#### id

```
template<typename _CharT>
locale::id std::messages<_CharT>::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 6.689 std::messages\_base Struct Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages\_base:



### Public Types

- typedef int **catalog**

### 6.689.1 Detailed Description

Messages facet base class providing catalog typedef.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 6.690 std::messages\_byname< \_CharT > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages\_byname< \_CharT >:



## Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string< _CharT >` **string\_type**

## Public Member Functions

- **messages\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **messages\_byname** (const `string` & \_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- `string_type` **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const `string_type` & \_\_s) const
- catalog **open** (const `basic_string`< char > &, const `locale` &, const char \*) const
- catalog **open** (const `basic_string`< char > & \_\_s, const `locale` & \_\_loc) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- `string_type` **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const `string_type` & \_\_msg) const
- virtual void **do\_close** (catalog) const
- void **do\_close** (catalog) const
- void **do\_close** (catalog) const
- `string` **do\_get** (catalog, int, int, const `string` &) const
- virtual `string_type` **do\_get** (catalog, int, int, const `string_type` & \_\_default) const
- `wstring` **do\_get** (catalog, int, int, const `wstring` &) const
- virtual catalog **do\_open** (const `basic_string`< char > &, const `locale` &) const
- `messages`< wchar\_t >::catalog **do\_open** (const `basic_string`< char > &, const `locale` &) const
- `messages`< char >::catalog **do\_open** (const `basic_string`< char > &, const `locale` &) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char \* `_M_name_messages`

#### 6.690.1 Detailed Description

```
template<typename _CharT>
class std::messages_byname< _CharT >
```

class messages\_byname [22.2.7.2].

#### 6.690.2 Member Function Documentation

##### do\_get()

```
string std::messages< char >::do_get (
 catalog ,
 int ,
 int ,
 const string &) const [protected], [inherited]
```

Specializations for required instantiations.

#### 6.690.3 Member Data Documentation

##### id

```
template<typename _CharT>
locale::id std::messages< _CharT >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

### 6.691 std::minus< \_Tp > Struct Template Reference

```
#include <stl_function.h>
```



Inheritance diagram for std::minus<\_Tp>:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 6.691.1 Detailed Description

**template**<typename `_Tp`>  
**struct** `std::minus<_Tp>`

One of the [math functors](#).

#### 6.691.2 Member Typedef Documentation

##### **first\_argument\_type**

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::first\_argument\_type [inherited]  
 first\_argument\_type is the type of the first argument

##### **result\_type**

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::result\_type [inherited]  
 result\_type is the return type

##### **second\_argument\_type**

typedef `_Tp` [std::binary\\_function](#)< `_Tp`, `_Tp`, `_Tp` >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.692 std::minus< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::minus< void >:



### Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)
- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename `_Tp`, typename `_Up`>  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`_t`) - [std::forward](#)< `_Up` >(`_u`))) -> decltype([std::forward](#)< `_Tp` >(`_t`) - [std::forward](#)< `_Up` >(`_u`))

#### 6.692.1 Detailed Description

One of the [math functors](#).

#### 6.692.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef void std::binary_function< void, void, void >::first_argument_type
```

`first_argument_type` is the type of the first argument

##### `result_type`

```
typedef void std::binary_function< void, void, void >::result_type
```

`result_type` is the return type

##### `second_argument_type`

```
typedef void std::binary_function< void, void, void >::second_argument_type
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.693 `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >` Class Template Reference

```
#include <mod_based_range_hashing.hpp>
```

### Protected Types

- typedef `Size_Type` **size\_type**

### Protected Member Functions

- void **notify\_resized** (`size_type` s)
- `size_type` **range\_hash** (`size_type` s) const
- void **swap** ([mod\\_based\\_range\\_hashing](#) &other)

#### 6.693.1 Detailed Description

`template<typename Size_Type>`

`class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >`

Mod based range hashing.

The documentation for this class was generated from the following file:

- [mod\\_based\\_range\\_hashing.hpp](#)

## 6.694 `std::modulus< _Tp >` Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::modulus< _Tp >`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- constexpr **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

### 6.694.1 Detailed Description

```
template<typename _Tp>
struct std::modulus<_Tp>
```

One of the [math functors](#).

### 6.694.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.695 std::modulus< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::modulus< void >`:



## Public Types

- typedef void [first\\_argument\\_type](#)
- typedef `__is_transparent` [is\\_transparent](#)

- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) %  
[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) % [std::forward](#)< \_Up >(\_\_u))

#### 6.695.1 Detailed Description

One of the [math functors](#).

#### 6.695.2 Member Typedef Documentation

##### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### result\_type

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

##### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.696 std::money\_base Class Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::money_base`:



### Public Types

- enum { **\_S\_minus** , **\_S\_zero** , **\_S\_end** }
- enum **part** {  
  **none** , **space** , **symbol** , **sign** ,  
  **value** }

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**

#### 6.696.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

`moneypunct::pos_format()` and `moneypunct::neg_format()` for details of how these fields are interpreted.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 6.697 std::money\_get<\_CharT, \_InIter> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::money\_get<\_CharT, \_InIter>:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

### Public Member Functions

- [money\\_get](#) (size\_t \_\_refs=0)
- template<bool \_Intl>  
[\\_GLIBCXX\\_BEGIN\\_NAMESPACE\\_LDBL\\_OR\\_CXX11](#) `_M_extract` ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, [string](#) &\_\_units) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long double &\_\_units) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, [string\\_type](#) &\_\_digits) const

### Static Public Attributes

- static [locale::id](#) [id](#)

### Protected Member Functions

- virtual [~money\\_get](#) ()
- template<bool \_Intl>  
[iter\\_type](#) [\\_M\\_extract](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, [string](#) &\_\_digits) const
- virtual [iter\\_type](#) [do\\_get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long double &\_\_units) const
- virtual [iter\\_type](#) [do\\_get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, bool \_\_intl, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, [string\\_type](#) &\_\_digits) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### 6.697.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::money_get< _CharT, _InIter >
```

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

### 6.697.2 Member Typedef Documentation

#### `char_type`

```
template<typename _CharT, typename _InIter>
typedef _CharT std::money_get< _CharT, _InIter >::char_type
Public typedefs.
```

#### `iter_type`

```
template<typename _CharT, typename _InIter>
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
Public typedefs.
```

#### `string_type`

```
template<typename _CharT, typename _InIter>
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
Public typedefs.
```

### 6.697.3 Constructor & Destructor Documentation

#### `money_get()`

```
template<typename _CharT, typename _InIter>
std::money_get< _CharT, _InIter >::money_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).



**~money\_get()**

```
template<typename _CharT, typename _InIter>
virtual std::money_get< _CharT, _InIter >::~~money_get () [inline], [protected], [virtual]
Destructor.
```

**6.697.4 Member Function Documentation****do\_get()** [1/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __units) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#).

Referenced by [get\(\)](#), and [get\(\)](#).

**do\_get()** [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 string_type & __digits) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::widen\(\)](#).

**get()** [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
```

```
ios_base::iostate & __err,
long double & __units) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `money_punct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `money_punct::frac_digits()` \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

#### Parameters

|                      |                                                                              |
|----------------------|------------------------------------------------------------------------------|
| <code>__s</code>     | Start of characters to parse.                                                |
| <code>__end</code>   | End of characters to parse.                                                  |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;money_punct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                               |
| <code>__err</code>   | Error field to set if parsing fails.                                         |
| <code>__units</code> | Place to store result of parsing.                                            |

#### Returns

Iterator referencing first character beyond valid money amount.

References [do\\_get\(\)](#).

#### get() [2/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 string_type & __digits) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `money_punct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `digits`. For example, the string \$10.01 in a US locale would store 1001 in `digits`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

#### Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__s</code> | Start of characters to parse. |
|------------------|-------------------------------|

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__end</code>    | End of characters to parse.                                                 |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;money_punct&lt;CharT,intl&gt;&gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                              |
| <code>__err</code>    | Error field to set if parsing fails.                                        |
| <code>__digits</code> | Place to store result of parsing.                                           |

#### Returns

Iterator referencing first character beyond valid money amount.

References [do\\_get\(\)](#).

#### 6.697.5 Member Data Documentation

##### id

```
template<typename _CharT, typename _InIter>
locale::id std::money_get<_CharT, _InIter>::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

#### 6.698 `std::money_put<_CharT, _OutIter>` Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::money_put<_CharT, _OutIter>`:



#### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- `money_put` (`size_t __refs=0`)
- `template<bool _Intl>`  
`_OutIter _M_insert (iter_type __s, ios_base & __io, char_type __fill, const string_type & __digits) const`
- `iter_type put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const`
- `iter_type put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const`

## Static Public Attributes

- static `locale::id id`

## Protected Member Functions

- virtual `~money_put ()`
- `template<bool _Intl>`  
`iter_type _M_insert (iter_type __s, ios_base & __io, char_type __fill, const string_type & __digits) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const`
- virtual `iter_type do_put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const`

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

### 6.98.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::money_put< _CharT, _OutIter >
```

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

### 6.98.2 Member Typedef Documentation

#### `char_type`

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::money_put< _CharT, _OutIter >::char_type
```

Public typedefs.

#### `iter_type`

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

**string\_type**

```
template<typename _CharT, typename _OutIter>
typedef basic_string<_CharT> std::money_put<_CharT, _OutIter>::string_type
Public typedefs.
```

**6.698.3 Constructor & Destructor Documentation****money\_put()**

```
template<typename _CharT, typename _OutIter>
std::money_put<_CharT, _OutIter>::money_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

**~money\_put()**

```
template<typename _CharT, typename _OutIter>
virtual std::money_put<_CharT, _OutIter>::~~money_put () [inline], [protected], [virtual]
Destructor.
```

**6.698.4 Member Function Documentation****do\_put()** [1/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put<_CharT, _OutIter>::do_put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 const string_type & __digits) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

**See also**

`put()`.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__s</code>    | The stream to write to.                                                    |
| <code>__intl</code> | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> . |

|                       |                                            |
|-----------------------|--------------------------------------------|
| <code>__io</code>     | Source of facets and io state.             |
| <code>__fill</code>   | <code>char_type</code> to use for padding. |
| <code>__digits</code> | Place to store result of parsing.          |

**Returns**

Iterator after writing.

**do\_put()** [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 long double __units) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`. This function is a hook for derived classes to change the value returned.

**See also**

`put()`.

**Parameters**

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | The stream to write to.                                                     |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__fill</code>  | <code>char_type</code> to use for padding.                                  |
| <code>__units</code> | Place to store result of parsing.                                           |

**Returns**

Iterator after writing.

References `std::ios_base::getloc()`, `std::use_facet()`, and `std::__ctype_abstract_base<_CharT >::widen()`.  
Referenced by `put()`, and `put()`.

**put()** [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
```

```

iter_type __s,
bool __intl,
ios_base & __io,
char_type __fill,
const string_type & __digits) const [inline]

```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the string 1001 in a US locale would write \$10.01 to \_\_s.

This function works by returning the result of do\_put().

#### Parameters

|                       |                                                                            |
|-----------------------|----------------------------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                                    |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                             |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                                 |
| <code>__digits</code> | Place to store result of parsing.                                          |

#### Returns

Iterator after writing.

References [do\\_put\(\)](#).

#### put() [2/2]

```

template<typename _CharT, typename _OutIter>
iter_type std::money_put<_CharT, _OutIter >::put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 long double __units) const [inline]

```

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the value 1001 in a US locale would write \$10.01 to \_\_s.

This function works by returning the result of do\_put().

#### Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__s</code>    | The stream to write to.                                                    |
| <code>__intl</code> | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt;&gt;</code> . |
| <code>__io</code>   | Source of facets and io state.                                             |
| <code>__fill</code> | <code>char_type</code> to use for padding.                                 |

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__units</code> | Place to store result of parsing. |
|----------------------|-----------------------------------|

#### Returns

Iterator after writing.

References [do\\_put\(\)](#).

### 6.698.5 Member Data Documentation

#### id

```
template<typename _CharT, typename _OutIter>
locale::id std::money_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

### 6.699 std::moneyput< \_CharT, \_Intl > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::moneyput< _CharT, _Intl >`:



#### Public Types

- enum { **\_S\_minus** , **\_S\_zero** , **\_S\_end** }
- typedef `__moneyput_cache< _CharT, _Intl >` **\_\_cache\_type**
- enum **part** {  
**none** , **space** , **symbol** , **sign** ,  
**value** }



- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `moneypunct` (`__c_locale` `__cloc`, `const char *``__s`, `size_t` `__refs=0`)
- `moneypunct` (`__cache_type *``__cache`, `size_t` `__refs=0`)
- `moneypunct` (`size_t` `__refs=0`)
- `string_type` `curr_symbol` () const
- `char_type` `decimal_point` () const
- `int` `frac_digits` () const
- `string` `grouping` () const
- `string_type` `negative_sign` () const
- `string_type` `positive_sign` () const
- `char_type` `thousands_sep` () const
- pattern `pos_format` () const
- pattern `neg_format` () const

### Static Public Member Functions

- static pattern `_S_construct_pattern` (`char` `__precedes`, `char` `__space`, `char` `__posn`) throw ()

### Static Public Attributes

- static const `char *``_S_atoms`
- static const pattern `_S_default_pattern`
- static `locale::id` `id`
- static const `bool` `intl`

### Protected Member Functions

- virtual `~moneypunct` ()
- void `_M_initialize_moneypunct` (`__c_locale` `__cloc=0`, `const char *``__name=0`)
- void `_M_initialize_moneypunct` (`__c_locale`, `const char *`)
- void `_M_initialize_moneypunct` (`__c_locale`, `const char *`)
- void `_M_initialize_moneypunct` (`__c_locale`, `const char *`)
- void `_M_initialize_moneypunct` (`__c_locale`, `const char *`)
- virtual `string_type` `do_curr_symbol` () const
- virtual `char_type` `do_decimal_point` () const
- virtual `int` `do_frac_digits` () const
- virtual `string` `do_grouping` () const
- virtual pattern `do_neg_format` () const
- virtual `string_type` `do_negative_sign` () const
- virtual pattern `do_pos_format` () const
- virtual `string_type` `do_positive_sign` () const
- virtual `char_type` `do_thousands_sep` () const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### 6.699.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct< _CharT, _Intl >
```

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

### 6.699.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
Public typedefs.
```

#### string\_type

```
template<typename _CharT, bool _Intl>
typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type
Public typedefs.
```

### 6.699.3 Constructor & Destructor Documentation

#### moneypunct() [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### moneypunct() [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
 __cache_type * __cache,
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is an internal constructor.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| <code>__cache</code> | Cache for optimization.         |
| <code>__refs</code>  | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

**moneypunt()** [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct<_CharT, _Intl>::moneypunct (
 __c_locale __cloc,
 const char * __s,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__s</code>    | The name of a locale.           |
| <code>__refs</code> | Passed to the base facet class. |

References [std::locale::facet::facet\(\)](#).

**~moneypunt()**

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct<_CharT, _Intl>::~~moneypunt () [protected], [virtual]
```

Destructor.

**6.699.4 Member Function Documentation****curr\_symbol()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::curr_symbol () const [inline]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `returning moneypunct<char_↵_type>::do_curr_symbol()`.

**Returns**

*string\_type* representing a currency symbol.

References [do\\_curr\\_symbol\(\)](#).

**decimal\_point()**

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct<_CharT, _Intl>::decimal_point () const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning moneypunct<char_↵type>::do_decimal_point()`.

**Returns**

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_curr\_symbol()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected],
[virtual]
```

Return currency symbol string.

This function returns a *string\_type* to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[curr\\_symbol\(\)](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Referenced by [curr\\_symbol\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_frac\_digits()**

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const [inline], [protected],
[virtual]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[frac\\_digits\(\)](#) for details.

**Returns**

Number of digits in amount fraction.

Referenced by [frac\\_digits\(\)](#).

**do\_grouping()**

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct<_CharT, _Intl>::do_grouping () const [inline], [protected],
[virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_neg\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct<_CharT, _Intl>::do_neg_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[neg\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Referenced by [neg\\_format\(\)](#).

**do\_negative\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct<_CharT, _Intl>::do_negative_sign () const [inline], [protected],
[virtual]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

**See also**

[negative\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a negative sign.

Referenced by [negative\\_sign\(\)](#).

**do\_pos\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

**See also**

[pos\\_format\(\)](#) for details.

**Returns**

Pattern for money values.

Referenced by [pos\\_format\(\)](#).

**do\_positive\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual]
```

Return positive sign string.

This function returns a [string\\_type](#) to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

**See also**

[positive\\_sign\(\)](#) for details.

**Returns**

*string\_type* representing a positive sign.

Referenced by [positive\\_sign\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a [char\\_type](#) to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**frac\_digits()**

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const [inline]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning [std::moneypunct<char\\_type>::do\\_frac\\_digits\(\)](#).

The fractional part of a money amount is optional. But if it is present, there must be [frac\\_digits\(\)](#) digits.

**Returns**

Number of digits in amount fraction.

References [do\\_frac\\_digits\(\)](#).

**grouping()**

```
template<typename _CharT, bool _Intl>
string std::moneypunct<_CharT, _Intl>::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**neg\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::neg_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

**Returns**

Pattern for money values.

References [do\\_neg\\_format\(\)](#).

**negative\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::negative_sign () const [inline]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a negative sign.

References [do\\_negative\\_sign\(\)](#).

**pos\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

**Returns**

Pattern for money values.

References [do\\_pos\\_format\(\)](#).

**positive\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline]
```

Return positive sign string.

This function returns a *string\_type* to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

References [do\\_positive\\_sign\(\)](#).

**thousands\_sep()**

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a *char\_type* to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

**Returns**

*char\_type* representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).



### 6.699.5 Member Data Documentation

#### id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct<_CharT, _Intl>::id [static]
Numpunct facet id.
```

#### intl

```
template<typename _CharT, bool _Intl>
const bool std::moneypunct<_CharT, _Intl>::intl [static]
This value is provided by the standard, but no reason for its existence.
The documentation for this class was generated from the following file:
```

- [locale\\_facets\\_nonio.h](#)

## 6.700 std::moneypunct\_byname<\_CharT, \_Intl> Class Template Reference

```
#include <locale_facets_nonio.h>
Inheritance diagram for std::moneypunct_byname<_CharT, _Intl>:
```



### Public Types

- enum { **`_S_minus`** , **`_S_zero`** , **`_S_end`** }
- typedef `__moneypunct_cache<_CharT, _Intl>` **`__cache_type`**
- typedef `_CharT` **`char_type`**
- enum **`part`** {  
    **`none`** , **`space`** , **`symbol`** , **`sign`** ,  
    **`value`** }
- typedef `basic_string<_CharT>` **`string_type`**

### Public Member Functions

- **money\_punct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **money\_punct\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- string\_type **curr\_symbol** () const
- char\_type **decimal\_point** () const
- int **frac\_digits** () const
- [string](#) **grouping** () const
- string\_type **negative\_sign** () const
- string\_type **positive\_sign** () const
- char\_type **thousands\_sep** () const
  
- pattern **pos\_format** () const
- pattern **neg\_format** () const

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

### Protected Member Functions

- void **\_M\_initialize\_money\_punct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- void **\_M\_initialize\_money\_punct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_money\_punct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_money\_punct** (\_\_c\_locale, const char \*)
- void **\_M\_initialize\_money\_punct** (\_\_c\_locale, const char \*)
- virtual string\_type **do\_curr\_symbol** () const
- virtual char\_type **do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual [string](#) **do\_grouping** () const
- virtual pattern **do\_neg\_format** () const
- virtual string\_type **do\_negative\_sign** () const
- virtual pattern **do\_pos\_format** () const
- virtual string\_type **do\_positive\_sign** () const
- virtual char\_type **do\_thousands\_sep** () const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### 6.700.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct_byname< _CharT, _Intl >
```

class `moneypunct_byname` [22.2.6.4].

### 6.700.2 Member Function Documentation

#### `curr_symbol()`

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline], [inherited]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `returning moneypunct<char_↵_type>::do_curr_symbol()`.

#### Returns

*string\_type* representing a currency symbol.

References [do\\_curr\\_symbol\(\)](#).

#### `decimal_point()`

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning moneypunct<char_↵type>::do_decimal_point()`.

#### Returns

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

#### `do_curr_symbol()`

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

#### See also

`curr_symbol()` for details.

#### Returns

*string\_type* representing a currency symbol.

Referenced by [curr\\_symbol\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_frac\_digits()**

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

[frac\\_digits\(\)](#) for details.

**Returns**

Number of digits in amount fraction.

Referenced by [frac\\_digits\(\)](#).

**do\_grouping()**

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_neg\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

`neg_format()` for details.

Returns

Pattern for money values.

Referenced by [neg\\_format\(\)](#).

### **do\_negative\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

`negative_sign()` for details.

Returns

*string\_type* representing a negative sign.

Referenced by [negative\\_sign\(\)](#).

### **do\_pos\_format()**

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

`pos_format()` for details.

Returns

Pattern for money values.

Referenced by [pos\\_format\(\)](#).

### **do\_positive\_sign()**

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

`positive_sign()` for details.

Returns

*string\_type* representing a positive sign.

Referenced by [positive\\_sign\(\)](#).

### **do\_thousands\_sep()**

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

### **frac\_digits()**

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const [inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `returning moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

Returns

Number of digits in amount fraction.

References [do\\_frac\\_digits\(\)](#).

### **grouping()**

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

References [do\\_grouping\(\)](#).

**neg\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::neg_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

**Returns**

Pattern for money values.

References [do\\_neg\\_format\(\)](#).

**negative\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::negative_sign () const [inline], [inherited]
```

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char\_type>::do\_negative\_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg\_format() and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a negative sign.

References [do\\_negative\\_sign\(\)](#).

**pos\_format()**

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::pos_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

**Returns**

Pattern for money values.

References [do\\_pos\\_format\(\)](#).

**positive\_sign()**

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline], [inherited]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `returning moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

References [do\\_positive\\_sign\(\)](#).

**thousands\_sep()**

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `returning moneypunct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

**6.700.3 Member Data Documentation****id**

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**6.701 std::pmr::monotonic\_buffer\_resource Class Reference**

```
#include <memory_resource>
```



Inheritance diagram for std::pmr::monotonic\_buffer\_resource:



### Public Member Functions

- **monotonic\_buffer\_resource** (const monotonic\_buffer\_resource &)=delete
- **monotonic\_buffer\_resource** (memory\_resource \* \_\_upstream) noexcept
- **monotonic\_buffer\_resource** (size\_t \_\_initial\_size) noexcept
- **monotonic\_buffer\_resource** (size\_t \_\_initial\_size, memory\_resource \* \_\_upstream) noexcept
- **monotonic\_buffer\_resource** (void \* \_\_buffer, size\_t \_\_buffer\_size) noexcept
- **monotonic\_buffer\_resource** (void \* \_\_buffer, size\_t \_\_buffer\_size, memory\_resource \* \_\_upstream) noexcept
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const memory\_resource & \_\_other) const noexcept
- monotonic\_buffer\_resource & **operator=** (const monotonic\_buffer\_resource &)=delete
- void **release** () noexcept
- memory\_resource \* **upstream\_resource** () const noexcept

### Protected Member Functions

- void \* **do\_allocate** (size\_t \_\_bytes, size\_t \_\_alignment) override
- void **do\_deallocate** (void \*, size\_t, size\_t) override
- bool **do\_is\_equal** (const memory\_resource & \_\_other) const noexcept override

#### 6.701.1 Detailed Description

A memory resource that allocates from a fixed-size buffer.

The main feature of a `pmr::monotonic_buffer_resource` is that its `do_deallocate` does nothing. This makes it very fast because there is no need to manage a free list, and every allocation simply returns a new block of memory, rather than searching for a suitably-sized free block. Because deallocating is a no-op, the amount of memory used by the resource only grows until `release()` (or the destructor) is called to return all memory to upstream.

A `monotonic_buffer_resource` can be initialized with a buffer that will be used to satisfy all allocation requests, until the buffer is full. After that a new buffer will be allocated from the upstream resource. By using a stack buffer and `pmr::null_memory_resource()` as the upstream you can get a memory resource that only uses the stack and never dynamically allocates.

Since

C++17

## 6.701.2 Member Function Documentation

### do\_allocate()

```
void * std::pmr::monotonic_buffer_resource::do_allocate (
 size_t __bytes,
 size_t __alignment) [inline], [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

### do\_deallocate()

```
void std::pmr::monotonic_buffer_resource::do_deallocate (
 void * ,
 size_t ,
 size_t) [inline], [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

### do\_is\_equal()

```
bool std::pmr::monotonic_buffer_resource::do_is_equal (
 const memory_resource & __other) const [inline], [override], [protected], [virtual],
[noexcept]
```

Implements [std::pmr::memory\\_resource](#).

The documentation for this class was generated from the following file:

- [memory\\_resource](#)

## 6.702 std::move\_iterator<\_Iterator> Class Template Reference

```
#include <stl_iterator.h>
```

### Public Types

- using **difference\_type**
- using **iterator\_concept**
- using **iterator\_type**
- using **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- template<typename \_Iter>  
constexpr **move\_iterator** (const move\_iterator<\_Iter> & \_\_i)
- constexpr **move\_iterator** (iterator\_type \_\_i)
- constexpr iterator\_type **base** () &&
- constexpr const iterator\_type & **base** () const &noexcept
- constexpr reference **operator\*** () const
- constexpr move\_iterator **operator+** (difference\_type \_\_n) const
- constexpr move\_iterator & **operator++** ()
- constexpr void **operator++** (int)
- constexpr move\_iterator **operator++** (int)
- constexpr move\_iterator & **operator+=** (difference\_type \_\_n)
- constexpr move\_iterator **operator-** (difference\_type \_\_n) const
- constexpr move\_iterator & **operator--** ()

- constexpr move\_iterator **operator--** (int)
- constexpr move\_iterator & **operator-=** (difference\_type \_\_n)
- constexpr pointer **operator->** () const
- template<typename \_Iter>  
constexpr move\_iterator & **operator=** (const move\_iterator< \_Iter > &\_\_i)
- constexpr reference **operator[]** (difference\_type \_\_n) const

## Friends

- constexpr [iter\\_rvalue\\_reference\\_t](#)< \_Iterator > **iter\_move** (const move\_iterator &\_\_i) noexcept(noexcept(ranges::↵iter\_move(\_\_i.\_M\_current)))
- template<indirectly\_swappable< \_Iterator > \_Iter2>  
constexpr void **iter\_swap** (const move\_iterator &\_\_x, const move\_iterator< \_Iter2 > &\_\_y) noexcept(noexcept(ranges::↵iter\_swap(\_\_x.\_M\_current, \_\_y.\_M\_current)))
- template<sized\_sentinel\_for< \_Iterator > \_Sent>  
constexpr iter\_difference\_t< \_Iterator > **operator-** (const move\_iterator &\_\_x, const [move\\_sentinel](#)< \_Sent > &\_\_y)
- template<sized\_sentinel\_for< \_Iterator > \_Sent>  
constexpr iter\_difference\_t< \_Iterator > **operator-** (const [move\\_sentinel](#)< \_Sent > &\_\_x, const move\_iterator &\_\_y)
- template<sentinel\_for< \_Iterator > \_Sent>  
constexpr bool **operator==** (const move\_iterator &\_\_x, const [move\\_sentinel](#)< \_Sent > &\_\_y)

## 6.702.1 Detailed Description

```
template<typename _Iterator>
class std::move_iterator< _Iterator >
```

An iterator adaptor that yields an rvalue reference.

Class template move\_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 6.703 `std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>` Class Template Reference

```
#include <functional>
```

## Public Types

- using **result\_type**

## Public Member Functions

- [move\\_only\\_function](#) () noexcept
- template<typename \_Fn, typename \_Vt = decay\_t<\_Fn>>  
[move\\_only\\_function](#) (\_Fn &&\_\_f) noexcept(\_S\_nothrow\_init< \_Vt, \_Fn >())
- template<typename \_Tp, typename... \_Args>  
[move\\_only\\_function](#) (in\_place\_type\_t< \_Tp >, \_Args &&... \_\_args) noexcept(\_S\_nothrow\_init< \_Tp, \_Args... >())

- `template<typename _Tp, typename _Up, typename... _Args>`  
`move_only_function` (`in_place_type_t< _Tp >`, `initializer_list< _Up > __il`, `_Args &&... __args`) `noexcept(_S_↵`  
`nothrow_init< _Tp, initializer_list< _Up > &, _Args... >())`
- `move_only_function` (`move_only_function &&__x`) `noexcept`
- `move_only_function` (`nullptr_t`) `noexcept`
- `operator bool` () `const noexcept`
- `_Res operator()` (`_ArgTypes... __args`) `_GLIBCXX_MOF_CV noexcept(_Noex)`
- `template<typename _Fn>`  
`move_only_function & operator=` (`_Fn &&__f`) `noexcept(is_nothrow_constructible_v< move_only_function, _Fn`  
`>)`
- `move_only_function & operator=` (`move_only_function &&__x`) `noexcept`
- `move_only_function & operator=` (`nullptr_t`) `noexcept`
- `void swap` (`move_only_function &__x`) `noexcept`

## Friends

- `template<typename _Func>`  
`constexpr auto & __polyfunc::__base_of` (`_Func &`) `noexcept`
- `template<typename _Func>`  
`constexpr auto & __polyfunc::__invoker_of` (`_Func &`) `noexcept`
- `template<typename _Src, typename _Dst>`  
`constexpr bool __polyfunc::__is_invoker_convertible` () `noexcept`
- `bool operator==` (`const move_only_function &__x`, `nullptr_t`) `noexcept`
- `void swap` (`move_only_function &__x`, `move_only_function &__y`) `noexcept`

### 6.703.1 Detailed Description

`template<typename _Res, typename... _ArgTypes, bool _Noex>`  
`class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>`

Polymorphic function wrapper.

Since

C++23

The `std::move_only_function` class template is a call wrapper similar to `std::function`, but does not require the stored target function to be copyable.

It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `move_only_function::operator()` member function are respected when invoking the target function.

### 6.703.2 Constructor & Destructor Documentation

#### `move_only_function()` [1/6]

`template<typename _Res, typename... _ArgTypes, bool _Noex>`  
`std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function`  
`() [inline], [noexcept]`

Creates an empty object.

References `move_only_function()`.

Referenced by `move_only_function()`, `move_only_function()`, `move_only_function()`, `move_only_function()`, `move_only_function()`, `move_only_function()`, `operator=()`, `operator=()`, `operator=()`, `operator=()`, `operator==`, `swap()`, and `swap`.

#### `move_only_function()` [2/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
 nullptr_t) [inline], [noexcept]
```

Creates an empty object.

References [move\\_only\\_function\(\)](#).

#### `move_only_function()` [3/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
 move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.

References [move\\_only\\_function\(\)](#).

#### `move_only_function()` [4/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
 _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

#### `move_only_function()` [5/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
 in_place_type_t< _Tp > ,
 _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

#### `move_only_function()` [6/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
 in_place_type_t< _Tp > ,
 initializer_list< _Up > __il,
 _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.

References [move\\_only\\_function\(\)](#), and [std::forward\(\)](#).

### 6.703.3 Member Function Documentation

#### **operator bool()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

#### **operator()()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
 _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `move_only_function` specialization.

#### **Precondition**

Must not be empty.

References [std::forward\(\)](#), and [operator\(\)\(\)](#).

Referenced by [operator\(\)\(\)](#).

#### **operator=()** [1/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↵
Noex)>::operator= (
 _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References [move\\_only\\_function\(\)](#), [std::forward\(\)](#), and [operator=\(\)](#).

#### **operator=()** [2/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↵
Noex)>::operator= (
 move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.

References [move\\_only\\_function\(\)](#), [std::addressof\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

#### **operator=()** [3/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↵
Noex)>::operator= (
 nullptr_t) [inline], [noexcept]
```

Destroys the target object (if any).

References [move\\_only\\_function\(\)](#), and [operator=\(\)](#).

**swap()**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
 move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline], [noexcept]
```

Exchange the target objects (if any).

References [move\\_only\\_function\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#), and [swap](#).

**6.703.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
 const move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
 nullptr_t) [friend]
```

Check for emptiness by comparing with `nullptr`.

References [move\\_only\\_function\(\)](#), and [operator==](#).

Referenced by [operator==](#).

**swap**

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
 move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
 move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y)
[friend]
```

Exchange the target objects (if any).

References [move\\_only\\_function\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following file:

- [mofunc\\_impl.h](#)

**6.704 std::move\_sentinel<\_Sent> Class Template Reference**

```
#include <std_iterator.h>
```

**Public Member Functions**

- constexpr **move\_sentinel** (\_Sent \_\_s) noexcept(is\_nothrow\_move\_constructible\_v<\_Sent>)
- template<typename \_S2>  
constexpr **move\_sentinel** (const move\_sentinel<\_S2> &\_\_s) noexcept(is\_nothrow\_constructible\_v<\_Sent, const \_S2 &>)
- constexpr \_Sent **base** () const noexcept(is\_nothrow\_copy\_constructible\_v<\_Sent>)
- template<typename \_S2>  
constexpr move\_sentinel & **operator=** (const move\_sentinel<\_S2> &\_\_s) noexcept(is\_nothrow\_assignable\_v<\_Sent, const \_S2 &>)

**6.704.1 Detailed Description**

```
template<semiregular _Sent>
class std::move_sentinel<_Sent>
```

A sentinel adaptor for use with `std::move_iterator`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 6.705 `std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>` Class Template Reference

```
#include <multimap.h>
```

Inheritance diagram for `std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>`:



### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, multimap>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, multimap>` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair<const_Key, _Tp>` **value\_type**

### Public Member Functions

- **multimap** (`_Base_ref __x`)
- template<typename `_InputIterator`>  
**multimap** (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator`>  
**multimap** (`_InputIterator __first, _InputIterator __last, const allocator_type &__a`)
- **multimap** (`const _Compare &__comp, const _Allocator &__a=_Allocator()`)
- **multimap** (`const allocator_type &__a`)
- **multimap** (`const multimap &`)=default
- **multimap** (`const multimap &__m, const __type_identity_t<allocator_type> &__a`)



- **multimap** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_c=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **multimap** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a)
- **multimap** (multimap &&)=default
- **multimap** (multimap &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(noexcept([Base](#)([std::move](#)(\_\_m), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_position)
- template<\_\_heterogeneous\_tree\_key< multimap > \_Kt>  
size\_type **erase** (\_Kt &&\_\_x)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (iterator \_\_position)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (\_Pair &&\_\_x)
- iterator **insert** (const value\_type &\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_list)
- iterator **insert** (value\_type &&\_\_x)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- multimap & **operator=** (const multimap &)=default
- multimap & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- multimap & **operator=** (multimap &&)=default
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rend** () const noexcept

- reverse\_iterator **rend** () noexcept
- void **swap** (multimap &\_\_x) noexcept(*/\*conditional \*/*)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_Safe\_iterator

### 6.705.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator =
std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [multimap.h](#)

### 6.706 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc > Class Template Reference

```
#include <map>
```

#### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Tp **mapped\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Rep\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef [std::pair](#)< const \_Key, \_Tp > **value\_type**

#### Public Member Functions

- [multimap](#) ()=default
- template<typename \_InputIterator>  
[multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
[multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a↵  
a=allocator\_type())

- template<typename \_InputIterator>  
multimap (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
  - multimap (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
  - multimap (const allocator\_type &\_\_a)
  - multimap (const multimap &)=default
  - multimap (const multimap &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
  - multimap (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
  - multimap (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
  - multimap (multimap &&)=default
  - multimap (multimap &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value && \_Alloc\_traits::\_S\_always\_equal())
  - ~multimap ()=default
  - const\_iterator begin () const noexcept
  - iterator begin () noexcept
  - const\_iterator cbegin () const noexcept
  - const\_iterator cend () const noexcept
  - void clear () noexcept
  - const\_reverse\_iterator crbegin () const noexcept
  - const\_reverse\_iterator crend () const noexcept
  - template<typename... \_Args>  
iterator emplace (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator emplace\_hint (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool empty () const noexcept
  - const\_iterator end () const noexcept
  - iterator end () noexcept
  - size\_type erase (const key\_type &\_\_x)
  - iterator erase (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type get\_allocator () const noexcept
  - template<typename \_InputIterator>  
void insert (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void insert (initializer\_list< value\_type > \_\_l)
  - key\_compare key\_comp () const
  - size\_type max\_size () const noexcept
  - multimap & operator= (const multimap &)=default
  - multimap & operator= (initializer\_list< value\_type > \_\_l)
  - multimap & operator= (multimap &&)=default
  - const\_reverse\_iterator rbegin () const noexcept
  - reverse\_iterator rbegin () noexcept
  - const\_reverse\_iterator rend () const noexcept
  - reverse\_iterator rend () noexcept
  - size\_type size () const noexcept
  - void swap (multimap &\_\_x) noexcept(/\*conditional \*/)
  - value\_compare value\_comp () const
- 
- iterator insert (const value\_type &\_\_x)
  - iterator insert (value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< is\_constructible< value\_type, \_Pair >::value, iterator > insert (\_Pair &&\_\_x)

- iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_Pair>  
\_\_enable\_if\_t< `is_constructible`< value\_type, \_Pair && >::value, iterator > `insert` (const\_iterator \_\_position, \_Pair &&\_\_x)
- iterator `erase` (const\_iterator \_\_position)
- iterator `erase` (iterator \_\_position)
- iterator `find` (const key\_type &\_\_x)
- const\_iterator `find` (const key\_type &\_\_x) const
- size\_type `count` (const key\_type &\_\_x) const
- bool `contains` (const key\_type &\_\_x) const
- template<typename \_Kt>  
auto `contains` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- iterator `lower_bound` (const key\_type &\_\_x)
- const\_iterator `lower_bound` (const key\_type &\_\_x) const
- iterator `upper_bound` (const key\_type &\_\_x)
- const\_iterator `upper_bound` (const key\_type &\_\_x) const
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)
- `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const

## Friends

- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1>  
bool `operator`< (const `multimap`< \_K1, \_T1, \_C1, \_A1 > &, const `multimap`< \_K1, \_T1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_T1, typename \_C1, typename \_A1>  
bool `operator`== (const `multimap`< \_K1, \_T1, \_C1, \_A1 > &, const `multimap`< \_K1, \_T1, \_C1, \_A1 > &)

### 6.706.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>>
class std::multimap<_Key, _Tp, _Compare, _Alloc>
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                    |
| <code>_Tp</code>      | Type of mapped objects.                                                                 |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .            |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 6.706.2 Constructor & Destructor Documentation

#### `multimap()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap () [default]
```

Default constructor creates no elements.

#### `multimap()` [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a `multimap` with no elements.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

#### `multimap()` [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 const multimap< _Key, _Tp, _Compare, _Alloc > &) [default]
```

Multimap copy constructor.

Whether the allocator is copied depends on the allocator traits.

#### **multimap()** [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 multimap< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Multimap move constructor.

The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

#### **multimap()** [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from an initializer\_list.

#### **Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multimap consisting of copies of the elements from the initializer\_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

#### **multimap()** [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

#### **multimap()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __m,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

#### multimap() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 multimap<_Key, _Tp, _Compare, _Alloc> && __m,
 const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

#### multimap() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

#### multimap() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

#### multimap() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a multimap from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

#### multimap() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 __InputIterator __first,
 __InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from a range.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

#### ~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::~~multimap () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### 6.706.3 Member Function Documentation

#### begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

#### begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

#### cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.



**cend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

**clear()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear () [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::multimap<_Key, _Tp, _Compare, _Alloc>::contains (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
| <code>__x</code>                                                                    |                                          |

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
bool std::multimap<_Key, _Tp, _Compare, _Alloc>::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
| <code>__x</code>                                                                    |                                          |

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|            |                                          |
|------------|------------------------------------------|
| <b>__x</b> | Key of (key, value) pairs to be located. |
|------------|------------------------------------------|

**Returns**

Number of elements with specified key.

**crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Build and insert a `std::pair` into the multimap.

**Parameters**

|               |                                                                                                                                                        |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>__args</b> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Builds and inserts a std::pair into the multimap.

**Parameters**

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap<_Key, _Tp, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the multimap is empty.

**end() [1/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

**end() [2/2]**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

### equal\_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↵</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

### equal\_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::↵
equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↵</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

#### Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

### erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

#### Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__key</code> | Key of element to be erased. |
| <code>__x</code>   |                              |

#### Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multimap.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased .  |

#### Returns

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a multimap.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from a multimap.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a multimap.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a multimap.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

#### get\_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Get a copy of the memory allocation object.

#### insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

#### insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::multimap< _Key, _Tp,
_Tp, _Compare, _Alloc >::insert (
 _Pair && __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <a href="#"><code>_↔</code></a> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
| <a href="#"><code>_x</code></a> |                                                                                   |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                                 |                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------|
| <a href="#"><code>_↔</code></a> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
| <a href="#"><code>_x</code></a> |                                                                                   |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Referenced by [std::multimap< \\_Key, \\_Tp, \\_Cmp, polymorphic\\_allocator< pair< const \\_Key, \\_Tp > > >::insert\(\)](#).

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::multimap< _Key,
_Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 _Pair && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                                         |                                                                                   |
|-----------------------------------------|-----------------------------------------------------------------------------------|
| <a href="#"><code>__position</code></a> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <a href="#"><code>__x</code></a>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |



**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>        | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of `std::pairs` into the multimap.

**Parameters**

|   |                                                                                 |
|---|---------------------------------------------------------------------------------|
| ↔ | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
| ↔ |                                                                                 |
| ↔ |                                                                                 |
| ↔ |                                                                                 |
| / |                                                                                 |

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|   |                                                                                   |
|---|-----------------------------------------------------------------------------------|
| ↔ | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
| ↔ |                                                                                   |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

**key\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::multimap<_Key, _Tp, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

**lower\_bound()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::max_size () const [inline], [noexcept]
```

Returns the maximum size of the multimap.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
multimap & std::multimap<_Key, _Tp, _Compare, _Alloc>::operator= (
 const multimap<_Key, _Tp, _Compare, _Alloc> &) [default]
```

Multimap assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Multimap list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a multimap with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
 multimap< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

**rbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

**rend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**rend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
reverse_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

**size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::size () const [inline], [noexcept]
```

Returns the size of the multimap.

**swap()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
void std::multimap<_Key, _Tp, _Compare, _Alloc>::swap (
 multimap<_Key, _Tp, _Compare, _Alloc> &__x) [inline], [noexcept]
```

Swaps data with another multimap.

**Parameters**

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multimap of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

**upper\_bound()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
```

```
iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::upper_bound (
 const key_type &__x) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed. The documentation for this class was generated from the following files:

- [stl\\_map.h](#)
- [stl\\_multimap.h](#)

**6.707 std::multiplies<\_Tp> Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::multiplies<_Tp>`:

**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

**6.707.1 Detailed Description**

```
template<typename _Tp>
struct std::multiplies< _Tp >
```

One of the [math functors](#).

**6.707.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**6.708 std::multiplies< void > Struct Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::multiplies< void >:

**Public Types**

- typedef void [first\\_argument\\_type](#)
- typedef \_\_is\_transparent [is\\_transparent](#)

- typedef void [result\\_type](#)
- typedef void [second\\_argument\\_type](#)

### Public Member Functions

- template<typename \_Tp, typename \_Up>  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept([std::forward](#)< \_Tp >(\_\_t) \*[std::forward](#)< \_Up >(\_\_u))) -> decltype([std::forward](#)< \_Tp >(\_\_t) \*[std::forward](#)< \_Up >(\_\_u))

#### 6.708.1 Detailed Description

One of the [math functors](#).

#### 6.708.2 Member Typedef Documentation

##### first\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::first\_argument\_type  
first\_argument\_type is the type of the first argument

##### result\_type

typedef void [std::binary\\_function](#)< void, void, void >::result\_type  
result\_type is the return type

##### second\_argument\_type

typedef void [std::binary\\_function](#)< void, void, void >::second\_argument\_type  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 6.709 std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator > Class Template Reference

```
#include <multiset.h>
```

Inheritance diagram for std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >:



### Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, multiset > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**



- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::Safe\\_iterator](#)< \_Base\_iterator, multiset > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

## Public Member Functions

- **multiset** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- **multiset** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- **multiset** (const allocator\_type &\_\_a)
- **multiset** (const multiset &)=default
- **multiset** (const multiset &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- **multiset** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a↵  
a=allocator\_type())
- **multiset** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a)
- **multiset** (multiset &&)=default
- **multiset** (multiset &&\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(noexcept(↵  
Base([std::move](#)(\_\_m), \_\_a)))
- const \_Base & **M\_base** () const noexcept
- \_Base & **M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_position)
- template<\_\_heterogeneous\_tree\_key< multiset > \_Kt>  
size\_type **erase** (\_Kt &&\_\_x)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)

- iterator **erase** (const\_iterator \_\_position)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- void **insert** (initializer\_list< value\_type > \_\_l)
- iterator **insert** (value\_type &&\_\_x)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- multiset & **operator=** (const multiset &)=default
- multiset & **operator=** (initializer\_list< value\_type > \_\_l)
- multiset & **operator=** (multiset &&)=default
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- reverse\_iterator **rend** () noexcept
- void **swap** (multiset &\_\_x) noexcept(*/\*conditional \*/*)
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_IT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug:: \_Safe\_iterator

### 6.709.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>>

class std::\_\_debug::multiset< \_Key, \_Compare, \_Allocator >

Class std::multiset with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [multiset.h](#)

### 6.710 std::multiset< \_Key, \_Compare, \_Alloc > Class Template Reference

```
#include <set>
```

### Public Types

- typedef \_Alloc **allocator\_type**
- typedef \_Rep\_type::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Rep\_type::const\_reverse\_iterator **const\_reverse\_iterator**

- typedef \_Rep\_type::difference\_type **difference\_type**
- typedef \_Rep\_type::const\_iterator **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Rep\_type::const\_reverse\_iterator **reverse\_iterator**
- typedef \_Rep\_type::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

## Public Member Functions

- **multiset** ()=default
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator>  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- **multiset** (const \_Compare &\_\_comp, const allocator\_type &\_\_a=allocator\_type())
- **multiset** (const allocator\_type &\_\_a)
- **multiset** (const multiset &)=default
- **multiset** (const multiset &\_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a)
- **multiset** (initializer\_list< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **multiset** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- **multiset** (multiset &)=default
- **multiset** (multiset && \_\_m, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare >::value && \_Alloc\_traits::\_S\_always\_equal())
- **~multiset** ()=default
- iterator **begin** () const noexcept
- iterator **cbegin** () const noexcept
- iterator **end** () const noexcept
- void **clear** () noexcept
- reverse\_iterator **crbegin** () const noexcept
- reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_position)
- allocator\_type **get\_allocator** () const noexcept
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)

- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
  - void **insert** (initializer\_list< value\_type > \_\_l)
  - iterator **insert** (value\_type &&\_\_x)
  - key\_compare **key\_comp** () const
  - size\_type **max\_size** () const noexcept
  - multiset & **operator=** (const multiset &)=default
  - multiset & **operator=** (initializer\_list< value\_type > \_\_l)
  - multiset & **operator=** (multiset &&)=default
  - reverse\_iterator **rbegin** () const noexcept
  - reverse\_iterator **rend** () const noexcept
  - size\_type **size** () const noexcept
  - void **swap** (multiset &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare **value\_comp** () const
- 
- size\_type **count** (const key\_type &\_\_x) const
- 
- bool **contains** (const key\_type &\_\_x) const
  - template<typename \_Kt>  
auto **contains** (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x), void(), true)
- 
- iterator **find** (const key\_type &\_\_x)
  - const\_iterator **find** (const key\_type &\_\_x) const
- 
- iterator **lower\_bound** (const key\_type &\_\_x)
  - const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- 
- iterator **upper\_bound** (const key\_type &\_\_x)
  - const\_iterator **upper\_bound** (const key\_type &\_\_x) const
- 
- **std::pair**< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
  - **std::pair**< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const

## Friends

- template<typename \_K1, typename \_C1, typename \_A1>  
bool **operator**< (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)
- template<typename \_K1, typename \_C1, typename \_A1>  
bool **operator**== (const multiset< \_K1, \_C1, \_A1 > &, const multiset< \_K1, \_C1, \_A1 > &)

### 6.710.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >
```

A standard container made up of elements, which can be retrieved in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 6.710.2 Constructor & Destructor Documentation

#### `multiset()` [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset () [default]
```

Default constructor creates no elements.

#### `multiset()` [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
```

```
 const _Compare & __comp,
```

```
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a multiset with no elements.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

#### `multiset()` [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset (
```

```

 __InputIterator __first,
 __InputIterator __last) [inline]

```

Builds a multiset from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multiset consisting of copies of the elements from `[first,last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

#### **multiset()** [4/12]

```

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 __InputIterator __first,
 __InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]

```

Builds a multiset from a range.

#### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multiset consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

#### **multiset()** [5/12]

```

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 const multiset< _Key, _Compare, _Alloc > &) [default]

```

Multiset copy constructor.

Whether the allocator is copied depends on the allocator traits.

#### **multiset()** [6/12]

```

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 multiset< _Key, _Compare, _Alloc > &&) [default]

```

Multiset move constructor.

The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

### multiset() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from an initializer\_list.

#### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

### multiset() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

### multiset() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
 const multiset<_Key, _Compare, _Alloc> & __m,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

### multiset() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc>::multiset (
 multiset<_Key, _Compare, _Alloc> && __m,
 const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

**multiset()** [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

**multiset()** [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

**~multiset()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::~~multiset () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**6.710.3 Member Function Documentation****begin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

**cbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

**cend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.



**clear()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
void std::multiset<_Key, _Compare, _Alloc>::clear () [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
```

```
auto std::multiset<_Key, _Compare, _Alloc>::contains (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
bool std::multiset<_Key, _Compare, _Alloc>::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
size_type std::multiset<_Key, _Compare, _Alloc>::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

Number of elements with specified key.

**crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
template<typename... _Args>
iterator std::multiset<_Key, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

**Parameters**

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate the element instance to be inserted. |
|---------------------|-----------------------------------------------------------------|

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted. Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
template<typename... _Args>
iterator std::multiset<_Key, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element instance to be inserted.               |

#### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

#### `empty()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
bool std::multiset<_Key, _Compare, _Alloc>::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

#### `end()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
iterator std::multiset<_Key, _Compare, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

#### `equal_range()` [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::pair< iterator, iterator > std::multiset<_Key, _Compare, _Alloc>::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::pair< const_iterator, const_iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range
(
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                                    |                    |
|------------------------------------|--------------------|
| <code>↵</code><br><code>__x</code> | Key to be located. |
|------------------------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**erase()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                                    |                              |
|------------------------------------|------------------------------|
| <code>↵</code><br><code>__x</code> | Key of element to be erased. |
|------------------------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multiset.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a set.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a set.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>↵</code>   | Element to be located. |
| <code>__x</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the memory allocation object.

**insert()** [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _InputIterator>
void std::multiset< _Key, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that tries to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts an element into the multiset.

**Parameters**

|                 |                         |
|-----------------|-------------------------|
| <code>↔</code>  | Element to be inserted. |
| <code>_x</code> |                         |

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Referenced by [std::multiset< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#).

**insert()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Inserts an element into the multiset.

**Parameters**

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↔associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↔associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

**insert()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the multiset.

**Parameters**

|                |                                                                                    |
|----------------|------------------------------------------------------------------------------------|
| <code>↵</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>/</code> |                                                                                    |

Complexity similar to that of the range constructor.

**key\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object.

**lower\_bound()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>↵</code>  | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>↵</code>  | Key to be located. |
| <code>_x</code> |                    |



**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::multiset<_Key, _Compare, _Alloc>::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

**operator=() [1/3]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset<_Key, _Compare, _Alloc>::operator= (
 const multiset<_Key, _Compare, _Alloc> &) [default]
```

Multiset assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=() [2/3]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset<_Key, _Compare, _Alloc>::operator= (
 initializer_list<value_type> & __l) [inline]
```

Multiset list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↵ | An initializer_list. |
| ↵ |                      |
| ↵ |                      |
| ↵ |                      |
| / |                      |

This function fills a multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

**operator=() [3/3]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset & std::multiset<_Key, _Compare, _Alloc>::operator= (
 multiset<_Key, _Compare, _Alloc> &&) [default]
```

Move assignment operator.

**rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

### rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

### size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵_Key>>
```

```
size_type std::multiset< _Key, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the set.

### swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵_Key>>
```

```
void std::multiset< _Key, _Compare, _Alloc >::swap (
 multiset< _Key, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another multiset.

### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>↵_X</code> | A multiset of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

### upper\_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵_Key>>
```

```
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵_X</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp () const [inline]
```

Returns the comparison object.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

**6.711 \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag:



## Public Member Functions

- **multiway\_mergesort\_exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

### 6.711.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

### 6.711.2 Member Function Documentation

#### **\_\_get\_num\_threads()**

[\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads](#) () [inline], [inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

#### **set\_num\_threads()**

void [\\_\\_gnu\\_parallel::parallel\\_tag::set\\_num\\_threads](#) (  
[\\_ThreadIndex](#) \_\_num\_threads) [inline], [inherited]

Set the desired number of threads.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.712 [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#) Struct Reference

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



### Public Member Functions

- `multiway_mergesort_sampling_tag` ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)

#### 6.712.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

#### 6.712.2 Member Function Documentation

##### `__get_num_threads()`

[\\_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [inline], [inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

void `__gnu_parallel::parallel_tag::set_num_threads` (  
[\\_ThreadIndex](#) \_\_num\_threads) [inline], [inherited]

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.713 \_\_gnu\_parallel::multiway\_mergesort\_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_tag:



### Public Member Functions

- `multiway_mergesort_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 6.713.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

#### 6.713.2 Member Function Documentation

##### `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.714 std::mutex Class Reference

```
#include <mutex>
```

### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

### Public Member Functions

- **mutex** (const mutex &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- mutex & **operator=** (const mutex &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

#### 6.714.1 Detailed Description

The standard mutex type.

A simple, non-recursive, non-timed mutex.

Do not call `lock()` and `unlock()` directly, use a scoped lock type such as `std::unique_lock`, `std::lock_guard`, or (since C++17) `std::scoped_lock`.

Since

C++11

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 6.715 std::negate<\_Tp> Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::negate<_Tp>`:



### Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Tp [result\\_type](#)

## Public Member Functions

- constexpr **\_Tp operator()** (const \_Tp &\_\_x) const

### 6.715.1 Detailed Description

```
template<typename _Tp>
struct std::negate<_Tp>
```

One of the [math functors](#).

### 6.715.2 Member Typedef Documentation

#### argument\_type

```
typedef _Tp std::unary_function<_Tp, _Tp>::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Tp std::unary_function<_Tp, _Tp>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.716 std::negate< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::negate< void >:



## Public Types

- typedef void [argument\\_type](#)
- typedef \_\_is\_transparent **is\_transparent**
- typedef void [result\\_type](#)



**Public Member Functions**

- `template<typename _Tp>`  
`constexpr auto operator() (_Tp &&__t) const noexcept(noexcept(-std::forward<_Tp>(__t))) -> decltype(-std::forward<_Tp>(__t))`

**6.716.1 Detailed Description**

One of the [math functors](#).

**6.716.2 Member Typedef Documentation****argument\_type**

`typedef void std::unary_function< void, void >::argument_type`  
`argument_type` is the type of the argument

**result\_type**

`typedef void std::unary_function< void, void >::result_type`  
`result_type` is the return type

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**6.717 `std::negative_binomial_distribution<_IntType>` Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- `typedef _IntType result_type`

**Public Member Functions**

- **`negative_binomial_distribution`** (`_IntType __k, double __p=0.5`)
- **`negative_binomial_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomNumberGenerator>`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator>`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomNumberGenerator &__urng, const param\_type &__p)`
- `_IntType k () const`
- `result\_type max () const`
- `result\_type min () const`
- `template<typename _UniformRandomNumberGenerator>`  
`result\_type operator() (_UniformRandomNumberGenerator &__urng)`

- `template<typename _UniformRandomNumberGenerator>`  
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `double p () const`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::negative_binomial_distribution< _IntType1 > &__x)`
- `bool operator== (const negative_binomial_distribution &__d1, const negative_binomial_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::negative_binomial_distribution< _IntType1 > &__x)`

## 6.717.1 Detailed Description

`template<typename _IntType = int>`  
**class** `std::negative_binomial_distribution< _IntType >`

A `negative_binomial_distribution` random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Since

C++11

## 6.717.2 Member Typedef Documentation

### result\_type

`template<typename _IntType = int>`  
`typedef _IntType std::negative_binomial_distribution< _IntType >::result_type`  
 The type of the range of the distribution.

## 6.717.3 Member Function Documentation

### k()

`template<typename _IntType = int>`  
`_IntType std::negative_binomial_distribution< _IntType >::k () const [inline]`  
 Return the  $k$  parameter of the distribution.

### max()

`template<typename _IntType = int>`  
`result_type std::negative_binomial_distribution< _IntType >::max () const [inline]`  
 Returns the least upper bound value of the distribution.

### min()

`template<typename _IntType = int>`  
`result_type std::negative_binomial_distribution< _IntType >::min () const [inline]`  
 Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _↵
_IntType >::operator() (
 _UniformRandomNumberGenerator & __urng)
```

Generating functions.

**p()**

```
template<typename _IntType = int>
double std::negative_binomial_distribution< _IntType >::p () const [inline]
```

Return the  $p$  parameter of the distribution.

**param() [1/2]**

```
template<typename _IntType = int>
param_type std::negative_binomial_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

**6.717.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::negative_binomial_distribution< _IntType1 > & __x) [friend]
```

Inserts a negative\_binomial\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                   |
|-------------------|-------------------|
| <code>__os</code> | An output stream. |
|-------------------|-------------------|

|                  |                                                                           |
|------------------|---------------------------------------------------------------------------|
| <code>__x</code> | A <code>negative_binomial_distribution</code> random number distribution. |
|------------------|---------------------------------------------------------------------------|

### Returns

The output stream with the state of `__x` inserted or in an error state.

### `operator==`

```
template<typename _IntType = int>
bool operator== (
 const negative_binomial_distribution< _IntType > & __d1,
 const negative_binomial_distribution< _IntType > & __d2) [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

### `operator>>`

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::negative_binomial_distribution< _IntType1 > & __x) [friend]
```

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

### Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                              |
| <code>__x</code>  | A <code>negative_binomial_distribution</code> random number generator engine. |

### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.718 `std::nested_exception` Class Reference

```
#include <exception>
```

### Public Member Functions

- [nested\\_exception](#) () noexcept
- [nested\\_exception](#) (const [nested\\_exception](#) &) noexcept=default
- [exception\\_ptr](#) [nested\\_ptr](#) () const noexcept
- [nested\\_exception](#) & [operator=](#) (const [nested\\_exception](#) &) noexcept=default
- void [rethrow\\_nested](#) () const

### 6.718.1 Detailed Description

Mixin class that stores the current exception.

This type can be used via `std::throw_with_nested` to store the current exception nested within another exception.

Since

C++11

See also

`std::throw_with_nested`

### 6.718.2 Constructor & Destructor Documentation

#### `nested_exception()`

`std::nested_exception::nested_exception () [inline], [noexcept]`

The default constructor stores the current exception (if any).

References [std::current\\_exception\(\)](#).

### 6.718.3 Member Function Documentation

#### `nested_ptr()`

`exception_ptr std::nested_exception::nested_ptr () const [inline], [noexcept]`

Access the stored exception.

#### `rethrow_nested()`

`void std::nested_exception::rethrow_nested () const [inline]`

Rethrow the stored exception, or terminate if none was stored.

References [std::rethrow\\_exception\(\)](#), [rethrow\\_nested\(\)](#), and [std::terminate\(\)](#).

Referenced by [rethrow\\_nested\(\)](#).

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

## 6.719 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

`#include <throw_allocator.h>`

### 6.719.1 Detailed Description

Never enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.720 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

`#include <throw_allocator.h>`

### 6.720.1 Detailed Description

Never enter the condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.721 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

```
#include <ext/new_allocator.h>
```

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::true_type` **propagate\_on\_container\_move\_assignment**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- **new\_allocator** (const new\_allocator &) noexcept
- template<typename \_Tp1>  
  **new\_allocator** (const new\_allocator<\_Tp1> &) noexcept
- constexpr `_Tp *` **allocate** (size\_type \_\_n, const void \* =static\_cast<const void \* >(0))
- constexpr `std::allocation_result<_Tp *, size_t>` **allocate\_at\_least** (size\_t \_\_n)
- constexpr void **deallocate** (`_Tp *` \_\_p, size\_type \_\_n)

### Static Public Member Functions

- static constexpr void **\_S\_check\_allocation\_limit** (size\_t \_\_n)

#### 6.721.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::new_allocator<_Tp>
```

An allocator that uses global `new`, as per C++03 [20.4.1].  
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is a non-standard extension that can be used to guarantee allocation from `new` even if the library has been configured to use a different implementation for `std::allocator`.

#### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

The documentation for this class was generated from the following file:

- [ext/new\\_allocator.h](#)

## 6.722 `__gnu_pbds::detail::no_throw_copies< Key, Mapped >` Struct Template Reference

```
#include <types_traits.hpp>
```

### Public Types

- `typedef integral_constant< int, __simple > indicator`

### Static Public Attributes

- `static const bool __simple`

#### 6.722.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.723 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

```
#include <types_traits.hpp>
```

### Public Types

- `typedef integral_constant< int, is_simple< Key >::value > indicator`

### Static Public Attributes

- `static const bool __simple`

#### 6.723.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.724 `std::normal_distribution< _RealType >` Class Template Reference

```
#include <random>
```

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **normal\_distribution** (const [param\\_type](#) &\_\_p)
- **normal\_distribution** ([result\\_type](#) \_\_mean, [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- `_RealType` **mean** () const
- [result\\_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator**() (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator`>  
[result\\_type](#) **operator**() (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- `_RealType` **stddev** () const

## Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_ostream`< `_CharT`, `_Traits` > & **operator<<** (`std::basic_ostream`< `_CharT`, `_Traits` > &\_\_os, const `std::normal_distribution`< `_RealType1` > &\_\_x)
- template<typename `_RealType1`>  
bool **operator==** (const `std::normal_distribution`< `_RealType1` > &\_\_d1, const `std::normal_distribution`< `_RealType1` > &\_\_d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits`>  
`std::basic_istream`< `_CharT`, `_Traits` > & **operator>>** (`std::basic_istream`< `_CharT`, `_Traits` > &\_\_is, `std::normal_distribution`< `_RealType1` > &\_\_x)

### 6.724.1 Detailed Description

**template<typename `_RealType` = double>**  
**class `std::normal_distribution`< `_RealType` >**

A normal continuous distribution for random numbers.  
 The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$



Since

C++11

### 6.724.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.724.3 Constructor & Destructor Documentation

#### normal\_distribution()

```
template<typename _RealType = double>
std::normal_distribution< _RealType >::normal_distribution (
 result_type __mean,
 result_type __stddev = result_type(1)) [inline], [explicit]
```

Constructs a normal distribution with parameters *mean* and standard deviation.

### 6.724.4 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### mean()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::mean () const [inline]
```

Returns the mean of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::normal_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### operator>() [2/2]

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator()
```

```
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

References [std::log\(\)](#), and [std::sqrt\(\)](#).

#### **param()** [1/2]

```
template<typename _RealType = double>
param_type std::normal_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### **param()** [2/2]

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### **Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

#### **reset()**

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

#### **stddev()**

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::stddev () const [inline]
```

Returns the standard deviation of the distribution.

### **6.724.5 Friends And Related Symbol Documentation**

#### **operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::normal_distribution< _RealType1 > & __x) [friend]
```

Inserts a normal\_distribution random number distribution `__x` into the output stream `__os`.

#### **Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <code>__os</code> | An output stream.                                 |
| <code>__x</code>  | A normal_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
template<typename _RealType1>
bool operator== (
 const std::normal_distribution< _RealType1 > & __d1,
 const std::normal_distribution< _RealType1 > & __d2) [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::normal_distribution< _RealType1 > & __x) [friend]
```

Extracts a normal\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <code>__is</code> | An input stream.                                      |
| <code>__x</code>  | A normal_distribution random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.725 std::not\_equal\_to<\_Tp> Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::not_equal_to<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 6.725.1 Detailed Description

```
template<typename _Tp>
struct std::not_equal_to<_Tp>
```

One of the [comparison functors](#).

#### 6.725.2 Member Typedef Documentation

##### **first\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### **result\_type**

```
typedef bool std::binary_function<_Tp, _Tp, bool>::result_type [inherited]
result_type is the return type
```

##### **second\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.726 std::ranges::not\_equal\_to Struct Reference

```
#include <ranges_cmp.h>
```

### Public Types

- using `is_transparent`

### Static Public Member Functions

- `template<typename _Tp, typename _Up>`  
`static constexpr bool operator() (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval<_Tp>()==std::declval<_Up>()))`

#### 6.726.1 Detailed Description

`ranges::not_equal_to` function object type.

The documentation for this struct was generated from the following file:

- [ranges\\_cmp.h](#)

## 6.727 std::not\_equal\_to< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::not_equal_to< void >`:



### Public Types

- `typedef void first_argument_type`
- `typedef __is_transparent is_transparent`
- `typedef bool result_type`
- `typedef void second_argument_type`

### Public Member Functions

- `template<typename _Tp, typename _Up>`  
`constexpr auto operator() (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward<_Tp>(__t) !=std::forward<_Up>(__u))) -> decltype(std::forward<_Tp>(__t) !=std::forward<_Up>(__u))`

### 6.727.1 Detailed Description

One of the [comparison functors](#).

### 6.727.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

first\_argument\_type is the type of the first argument

#### result\_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

result\_type is the return type

#### second\_argument\_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 6.728 \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 > Struct Template Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 >:



### 6.728.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
```

A null node updatator, indicating that no node updates are required.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.729 `__gnu_pbds::null_type` Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::null_type`:



### 6.729.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.730 `std::experimental::fundamentals_v1::nullopt_t` Struct Reference

```
#include <optional>
```

## Public Types

- enum class `_Construct` { `_Token` }

## Public Member Functions

- constexpr `nullopt_t` (`_Construct`)

### 6.730.1 Detailed Description

Tag type to disengage optional objects.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

### 6.731 `std::num_get<_CharT, _InIter >` Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for `std::num_get<_CharT, _InIter >`:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)

## Public Member Functions

- [num\\_get](#) (`size_t` \_\_refs=0)
- template<typename `_ValueT`>  
`_InIter` [\\_M\\_extract\\_int](#) (`_InIter` \_\_beg, `_InIter` \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, `_ValueT` &\_\_v) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, bool &\_\_v) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, void \*&\_\_v) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, long &\_\_v) const
- [iter\\_type](#) [get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, unsigned short &\_\_v) const



- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v) const`

### Static Public Attributes

- static `locale::id id`

### Protected Member Functions

- virtual `~num_get ()`
- `iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT> iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2> __gnu_cxx::enable_if<__is_char<_CharT2>>::__value, int >::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2> __gnu_cxx::enable_if<!__is_char<_CharT2>>::__value, int >::__type _M_find (const _CharT2 * __zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

### 6.731.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >
```

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

### 6.731.2 Member Typedef Documentation

#### `char_type`

```
template<typename _CharT, typename _InIter>
typedef _CharT std::num_get< _CharT, _InIter >::char_type
Public typedefs.
```

#### `iter_type`

```
template<typename _CharT, typename _InIter>
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
Public typedefs.
```

### 6.731.3 Constructor & Destructor Documentation

#### `num_get()`

```
template<typename _CharT, typename _InIter>
std::num_get< _CharT, _InIter >::num_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

#### `~num_get()`

```
template<typename _CharT, typename _InIter>
virtual std::num_get< _CharT, _InIter >::~num_get () [inline], [protected], [virtual]
Destructor.
```

### 6.731.4 Member Function Documentation

#### `do_get()` [1/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
```

```
ios_base::iostate & __err,
long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### do\_get() [2/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get<_CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [3/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned int & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [4/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### do\_get() [5/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get<_CharT, _InIter>::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

**do\_get()** [6/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned short & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**do\_get()** [7/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 bool & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <code>__beg</code> | Start of input stream. |
| <code>__end</code> | End of input stream.   |

|                    |                             |
|--------------------|-----------------------------|
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::boolalpha](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::flags\(\)](#), and [std::ios\\_base::goodbit](#).

Referenced by [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#), and [std::num\\_get<\\_CharT, istreambuf\\_iterator<\\_CharT, \\_Traits > >::get\(\)](#).

**do\_get()** [8/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get<_CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 double & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc >::r](#)

**do\_get()** [9/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 float & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::r](#)

**do\_get()** [10/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

[get\(\)](#) for more details.

**Parameters**

|                    |                        |
|--------------------|------------------------|
| <code>__beg</code> | Start of input stream. |
| <code>__end</code> | End of input stream.   |



|                    |                             |
|--------------------|-----------------------------|
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::c\\_str\(\)](#), [std::ios\\_base::eofbit](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::r](#)

**do\_get()** [11/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get<_CharT, _InIter>::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 void *& __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

References [std::ios\\_base::basefield](#), [std::ios\\_base::flags\(\)](#), and [std::ios\\_base::hex](#).

**get()** [1/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get<_CharT, _InIter>::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
```

```

 ios_base::iostate & __err,
 bool & __v) const [inline]

```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Referenced by `std::basic_istream< char_type, traits_type >::operator>>()`, and `std::basic_istream< _CharT, _Traits >::operator>>()`.

#### get() [2/11]

```

template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 double & __v) const [inline]

```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |

|                    |                             |
|--------------------|-----------------------------|
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [3/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 float & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered.

Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [4/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### `get()` [5/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` `g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |

|                  |                             |
|------------------|-----------------------------|
| <code>__v</code> | Value to format and insert. |
|------------------|-----------------------------|

**Returns**

Iterator after reading.

**get()** [6/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [7/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
```

```
ios_base::iostate & __err,
unsigned int & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### `get()` [8/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [9/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [10/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned short & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

**get()** [11/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 void *& __v) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` `p` specifier.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.



If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

#### 6.731.5 Member Data Documentation

##### id

```
template<typename _CharT, typename _InIter>
locale::id std::num_get<_CharT, _InIter >::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

#### 6.732 std::num\_put<\_CharT, \_OutIter > Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for `std::num_put<_CharT, _OutIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Outlter` `iter_type`

## Public Member Functions

- `num_put` (`size_t` \_\_refs=0)
- template<typename `_ValueT`>  
`_Outlter` **`_M_insert_float`** (`_Outlter` \_\_s, `ios_base` &\_\_io, `_CharT` \_\_fill, `char` \_\_mod, `_ValueT` \_\_v) const
- template<typename `_ValueT`>  
`_Outlter` **`_M_insert_int`** (`_Outlter` \_\_s, `ios_base` &\_\_io, `_CharT` \_\_fill, `_ValueT` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `bool` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `const void *` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long long` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `double` \_\_v) const
- `iter_type put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long double` \_\_v) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~num_put` ()
- void **`_M_group_float`** (`const char *` \_\_grouping, `size_t` \_\_grouping\_size, `char_type` \_\_sep, `const char_type *` \_\_p, `char_type *` \_\_new, `char_type *` \_\_cs, `int` &\_\_len) const
- void **`_M_group_int`** (`const char *` \_\_grouping, `size_t` \_\_grouping\_size, `char_type` \_\_sep, `ios_base` &\_\_io, `char_type *` \_\_new, `char_type *` \_\_cs, `int` &\_\_len) const
- template<typename `_ValueT`>  
`iter_type` **`_M_insert_float`** (`iter_type`, `ios_base` &\_\_io, `char_type` \_\_fill, `char` \_\_mod, `_ValueT` \_\_v) const
- template<typename `_ValueT`>  
`iter_type` **`_M_insert_int`** (`iter_type`, `ios_base` &\_\_io, `char_type` \_\_fill, `_ValueT` \_\_v) const
- void **`_M_pad`** (`char_type` \_\_fill, `streamsize` \_\_w, `ios_base` &\_\_io, `char_type *` \_\_new, `const char_type *` \_\_cs, `int` &\_\_len) const
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `bool` \_\_v) const
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long` \_\_v) const
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long` \_\_v) const
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `long long` \_\_v) const
- virtual `iter_type do_put` (`iter_type` \_\_s, `ios_base` &\_\_io, `char_type` \_\_fill, `unsigned long long` \_\_v) const
- virtual `iter_type do_put` (`iter_type`, `ios_base` &, `char_type`, `double`) const
- virtual `iter_type do_put` (`iter_type`, `ios_base` &, `char_type`, `long double`) const
- virtual `iter_type do_put` (`iter_type`, `ios_base` &, `char_type`, `const void *`) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 6.732.1 Detailed Description

template<typename \_CharT, typename \_OutIter>  
class std::num\_put< \_CharT, \_OutIter >

Primary class template num\_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The num\_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_put facet.

#### 6.732.2 Member Typedef Documentation

##### char\_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::num_put< _CharT, _OutIter >::char_type
```

Public typedefs.

##### iter\_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

#### 6.732.3 Constructor & Destructor Documentation

##### num\_put()

```
template<typename _CharT, typename _OutIter>
std::num_put< _CharT, _OutIter >::num_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

##### ~num\_put()

```
template<typename _CharT, typename _OutIter>
virtual std::num_put< _CharT, _OutIter >::~num_put () [inline], [protected], [virtual]
```

Destructor.

## 6.732.4 Member Function Documentation

### do\_put() [1/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 bool __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                                                                                                                       |                               |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------|
| <a href="#"> <code>__s</code></a>    | Stream to write to.           |
| <a href="#"> <code>__io</code></a>   | Source of locale and flags.   |
| <a href="#"> <code>__fill</code></a> | Char_type to use for filling. |
| <a href="#"> <code>__v</code></a>    | Value to format and insert.   |

#### Returns

Iterator after writing.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::adjustfield](#), [std::ios\\_base::boolalpha](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::left](#), and [std::ios\\_base::width\(\)](#).

Referenced by [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#), and [std::num\\_put< \\_CharT, ostreambuf\\_iterator< \\_CharT, \\_Traits > >::put\(\)](#).

### do\_put() [2/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                                                                                                                      |                     |
|----------------------------------------------------------------------------------------------------------------------|---------------------|
| <a href="#"> <code>__s</code></a> | Stream to write to. |
|----------------------------------------------------------------------------------------------------------------------|---------------------|

|                                     |                               |
|-------------------------------------|-------------------------------|
| <a href="#"><code>__io</code></a>   | Source of locale and flags.   |
| <a href="#"><code>__fill</code></a> | Char_type to use for filling. |
| <a href="#"><code>__v</code></a>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put() [3/8]**

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                                     |                               |
|-------------------------------------|-------------------------------|
| <a href="#"><code>__s</code></a>    | Stream to write to.           |
| <a href="#"><code>__io</code></a>   | Source of locale and flags.   |
| <a href="#"><code>__fill</code></a> | Char_type to use for filling. |
| <a href="#"><code>__v</code></a>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put() [4/8]**

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                                                                                                                          |                               |
|--------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| <a href="#"></a><br><code>__s</code>    | Stream to write to.           |
| <a href="#"></a><br><code>__io</code>   | Source of locale and flags.   |
| <a href="#"></a><br><code>__fill</code> | Char_type to use for filling. |
| <a href="#"></a><br><code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [5/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                                                                                                                            |                               |
|----------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| <a href="#"></a><br><code>__s</code>    | Stream to write to.           |
| <a href="#"></a><br><code>__io</code>   | Source of locale and flags.   |
| <a href="#"></a><br><code>__fill</code> | Char_type to use for filling. |
| <a href="#"></a><br><code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**do\_put()** [6/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const void * __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                                     |                               |
|-------------------------------------|-------------------------------|
| <a href="#"><code>__s</code></a>    | Stream to write to.           |
| <a href="#"><code>__io</code></a>   | Source of locale and flags.   |
| <a href="#"><code>__fill</code></a> | Char_type to use for filling. |
| <a href="#"><code>__v</code></a>    | Value to format and insert.   |

#### Returns

Iterator after writing.

References [std::ios\\_base::basefield](#), [std::ios\\_base::flags\(\)](#), [std::ios\\_base::hex](#), [std::ios\\_base::showbase](#), and [std::ios\\_base::uppercase](#).

#### do\_put() [7/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put<_CharT, _OutIter>::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

#### Parameters

|                                     |                               |
|-------------------------------------|-------------------------------|
| <a href="#"><code>__s</code></a>    | Stream to write to.           |
| <a href="#"><code>__io</code></a>   | Source of locale and flags.   |
| <a href="#"><code>__fill</code></a> | Char_type to use for filling. |
| <a href="#"><code>__v</code></a>    | Value to format and insert.   |

#### Returns

Iterator after writing.

**do\_put()** [8/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [1/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 bool __v) const [inline]
```

Numeric formatting.

Formats the boolean `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats `v` as an int.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |



**Returns**

Iterator after writing.

Referenced by [std::basic\\_ostream< \\_CharT, \\_Traits >::operator<<\(\)](#).

**put()** [2/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const void * __v) const [inline]
```

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats *v* as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [3/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 double __v) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both `fixed` and `scientific` are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numunct::decimal_point()`. Thousands separators are inserted according to `numunct::grouping()` and `numunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

### Returns

Iterator after writing.

### put() [4/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numunct::decimal_point()`. Thousands separators are inserted according to `numunct::grouping()` and `numunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

|                   |                             |
|-------------------|-----------------------------|
| <code>__s</code>  | Stream to write to.         |
| <code>__io</code> | Source of locale and flags. |

|                     |                               |
|---------------------|-------------------------------|
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [5/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long double __v) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [6/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                          |                               |
|--------------------------|-------------------------------|
| <code>↔<br/>_s</code>    | Stream to write to.           |
| <code>↔<br/>_io</code>   | Source of locale and flags.   |
| <code>↔<br/>_fill</code> | Char_type to use for filling. |
| <code>↔<br/>_v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**put()** [7/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

### Returns

Iterator after writing.

### put() [8/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long long __v) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

### Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | Stream to write to. |
|------------------|---------------------|

|                           |                               |
|---------------------------|-------------------------------|
| <code>_↔<br/>_io</code>   | Source of locale and flags.   |
| <code>_↔<br/>_fill</code> | Char_type to use for filling. |
| <code>_↔<br/>_v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

**6.732.5 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::num_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

**6.733 std::numeric\_limits< \_Tp > Struct Template Reference**

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< _Tp >`:

**Static Public Member Functions**

- static constexpr `_Tp` [denorm\\_min](#) () noexcept
- static constexpr `_Tp` [epsilon](#) () noexcept
- static constexpr `_Tp` [infinity](#) () noexcept
- static constexpr `_Tp` [lowest](#) () noexcept
- static constexpr `_Tp` [max](#) () noexcept
- static constexpr `_Tp` [min](#) () noexcept

- static constexpr \_Tp [quiet\\_NaN](#) () noexcept
- static constexpr \_Tp [round\\_error](#) () noexcept
- static constexpr \_Tp [signaling\\_NaN](#) () noexcept

### Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

### 6.733.1 Detailed Description

**template<typename \_Tp>**  
**struct std::numeric\_limits< \_Tp >**

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

### 6.733.2 Member Function Documentation

#### denorm\_min()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::denorm_min () [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is `false`, this is the minimum positive normalized value.

#### epsilon()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::epsilon () [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable. Referenced by `std::binomial_distribution< _IntType >::operator()()`, `std::poisson_distribution< _IntType >::operator()()`, and `std::operator<<()`.

**infinity()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if has\_infinity.

**lowest()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

Referenced by [std::cauchy\\_distribution<\\_RealType>::min\(\)](#), [std::extreme\\_value\\_distribution<\\_RealType>::min\(\)](#), [std::normal\\_distribution<\\_RealType>::min\(\)](#), and [std::student\\_t\\_distribution<\\_RealType>::min\(\)](#).

**max()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\(\)](#), [std::bernoulli\\_distribution::max\(\)](#), [std::cauchy\\_distribution<\\_RealType>::max\(\)](#), [std::chi\\_squared\\_distribution<\\_RealType>::max\(\)](#), [std::exponential\\_distribution<\\_RealType>::max\(\)](#), [std::extreme\\_value\\_distribution<\\_RealType>::max\(\)](#), [std::fisher\\_f\\_distribution<\\_RealType>::max\(\)](#), [std::gamma\\_distribution<\\_RealType>::max\(\)](#), [std::geometric\\_distribution<\\_IntType>::max\(\)](#), [std::lognormal\\_distribution<\\_RealType>::max\(\)](#), [std::negative\\_binomial\\_distribution<\\_IntType>::max\(\)](#), [std::normal\\_distribution<\\_RealType>::max\(\)](#), [std::poisson\\_distribution<\\_IntType>::max\(\)](#), [std::student\\_t\\_distribution<\\_RealType>::max\(\)](#), [std::weibull\\_distribution<\\_RealType>::max\(\)](#), [std::tr2::dynamic\\_bitset<\\_WordT, \\_Alloc>::max\\_size\(\)](#), [std::binomial\\_distribution<\\_IntType>::operator\(\)\(\)](#), [std::independent\\_bits\\_engine<\\_WordT, \\_Alloc>::max\\_size\(\)](#), and [std::operator<<\(\)](#).

**min()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Referenced by [std::bernoulli\\_distribution::min\(\)](#).

**quiet\_NaN()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if has\_quiet\_NaN.

**round\_error()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

**signaling\_NaN()**

```
template<typename _Tp>
constexpr _Tp std::numeric_limits<_Tp>::signaling_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.



### 6.733.3 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr], [inherited]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Referenced by `std::generate_canonical()`, `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator()`, and `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr], [inherited]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr], [inherited]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr], [inherited]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr], [inherited]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr], [inherited]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr], [inherited]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr], [inherited]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr], [inherited]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr], [inherited]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr], [inherited]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr], [inherited]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr], [inherited]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr], [inherited]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr], [inherited]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr], [inherited]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr], [inherited]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr], [inherited]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr], [inherited]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr], [inherited]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Referenced by [std::generate\\_canonical\(\)](#).

### round\_style

`float_round_style` `std::__numeric_limits_base::round_style` [static], [constexpr], [inherited]

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

### tinyness\_before

`bool` `std::__numeric_limits_base::tinyness_before` [static], [constexpr], [inherited]

True if tininess is detected before rounding. (see IEC 559)

### traps

`bool` `std::__numeric_limits_base::traps` [static], [constexpr], [inherited]

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

## 6.734 std::numeric\_limits< bool > Struct Reference

```
#include <limits>
```

Inheritance diagram for `std::numeric_limits< bool >`:



### Static Public Member Functions

- static constexpr `bool` **denorm\_min** () noexcept
- static constexpr `bool` **epsilon** () noexcept
- static constexpr `bool` **infinity** () noexcept
- static constexpr `bool` **lowest** () noexcept
- static constexpr `bool` **max** () noexcept
- static constexpr `bool` **min** () noexcept
- static constexpr `bool` **quiet\_NaN** () noexcept
- static constexpr `bool` **round\_error** () noexcept
- static constexpr `bool` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

#### 6.734.1 Detailed Description

numeric\_limits<bool> specialization.

### 6.734.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.735 std::numeric\_limits< char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< char >:

**Static Public Member Functions**

- static constexpr char **denorm\_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet\_NaN** () noexcept
- static constexpr char **round\_error** () noexcept
- static constexpr char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.735.1 Detailed Description

numeric\_limits<char> specialization.



### 6.735.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.736 std::numeric\_limits< char16\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< char16\_t >:

**Static Public Member Functions**

- static constexpr char16\_t **denorm\_min** () noexcept
- static constexpr char16\_t **epsilon** () noexcept
- static constexpr char16\_t **infinity** () noexcept
- static constexpr char16\_t **lowest** () noexcept
- static constexpr char16\_t **max** () noexcept
- static constexpr char16\_t **min** () noexcept
- static constexpr char16\_t **quiet\_NaN** () noexcept
- static constexpr char16\_t **round\_error** () noexcept
- static constexpr char16\_t **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.736.1 Detailed Description

numeric\_limits<char16\_t> specialization.

## 6.736.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.737 std::numeric\_limits< char32\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< char32\_t >:

**Static Public Member Functions**

- static constexpr char32\_t **denorm\_min** () noexcept
- static constexpr char32\_t **epsilon** () noexcept
- static constexpr char32\_t **infinity** () noexcept
- static constexpr char32\_t **lowest** () noexcept
- static constexpr char32\_t **max** () noexcept
- static constexpr char32\_t **min** () noexcept
- static constexpr char32\_t **quiet\_NaN** () noexcept
- static constexpr char32\_t **round\_error** () noexcept
- static constexpr char32\_t **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.737.1 Detailed Description

numeric\_limits<char32\_t> specialization.



### 6.737.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.738 std::numeric\_limits< double > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< double >:

**Static Public Member Functions**

- static constexpr double **denorm\_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet\_NaN** () noexcept
- static constexpr double **round\_error** () noexcept
- static constexpr double **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.738.1 Detailed Description

numeric\_limits<double> specialization.

### 6.738.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.739 std::numeric\_limits< float > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< float >:

**Static Public Member Functions**

- static constexpr float **denorm\_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet\_NaN** () noexcept
- static constexpr float **round\_error** () noexcept
- static constexpr float **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.739.1 Detailed Description

numeric\_limits<float> specialization.



### 6.739.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.740 std::numeric\_limits< int > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< int >:

**Static Public Member Functions**

- static constexpr int **denorm\_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet\_NaN** () noexcept
- static constexpr int **round\_error** () noexcept
- static constexpr int **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.740.1 Detailed Description

numeric\_limits<int> specialization.

### 6.740.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.741 std::numeric\_limits< long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< long >:

**Static Public Member Functions**

- static constexpr long **denorm\_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet\_NaN** () noexcept
- static constexpr long **round\_error** () noexcept
- static constexpr long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.741.1 Detailed Description

numeric\_limits<long> specialization.



### 6.741.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.742 std::numeric\_limits< long double > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< long double >:

**Static Public Member Functions**

- static constexpr long double **denorm\_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet\_NaN** () noexcept
- static constexpr long double **round\_error** () noexcept
- static constexpr long double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

#### 6.742.1 Detailed Description

numeric\_limits<long double> specialization.

### 6.742.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.743 std::numeric\_limits< long long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< long long >:

**Static Public Member Functions**

- static constexpr long long **denorm\_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet\_NaN** () noexcept
- static constexpr long long **round\_error** () noexcept
- static constexpr long long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.743.1 Detailed Description

numeric\_limits<long long> specialization.



### 6.743.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.744 std::numeric\_limits< short > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< short >:

**Static Public Member Functions**

- static constexpr short **denorm\_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet\_NaN** () noexcept
- static constexpr short **round\_error** () noexcept
- static constexpr short **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.744.1 Detailed Description

numeric\_limits<short> specialization.

### 6.744.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that *radix* raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that *radix* raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.745 std::numeric\_limits< signed char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< signed char >:

**Static Public Member Functions**

- static constexpr signed char **denorm\_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet\_NaN** () noexcept
- static constexpr signed char **round\_error** () noexcept
- static constexpr signed char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.745.1 Detailed Description

numeric\_limits<signed char> specialization.



### 6.745.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.746 std::numeric\_limits< unsigned char > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned char >:

**Static Public Member Functions**

- static constexpr unsigned char **denorm\_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet\_NaN** () noexcept
- static constexpr unsigned char **round\_error** () noexcept
- static constexpr unsigned char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.746.1 Detailed Description

numeric\_limits<unsigned char> specialization.

## 6.746.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.747 std::numeric\_limits< unsigned int > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned int >:

**Static Public Member Functions**

- static constexpr unsigned int **denorm\_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet\_NaN** () noexcept
- static constexpr unsigned int **round\_error** () noexcept
- static constexpr unsigned int **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.747.1 Detailed Description

numeric\_limits<unsigned int> specialization.



## 6.747.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.748 std::numeric\_limits< unsigned long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned long >:

**Static Public Member Functions**

- static constexpr unsigned long **denorm\_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet\_NaN** () noexcept
- static constexpr unsigned long **round\_error** () noexcept
- static constexpr unsigned long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.748.1 Detailed Description

numeric\_limits<unsigned long> specialization.

## 6.748.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.749 std::numeric\_limits< unsigned long long > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned long long >:

**Static Public Member Functions**

- static constexpr unsigned long long **denorm\_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet\_NaN** () noexcept
- static constexpr unsigned long long **round\_error** () noexcept
- static constexpr unsigned long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

#### 6.749.1 Detailed Description

numeric\_limits<unsigned long long> specialization.



### 6.749.2 Member Data Documentation

#### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

#### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

#### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

#### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

#### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

#### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

#### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

#### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

#### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

#### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.750 std::numeric\_limits< unsigned short > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< unsigned short >:

**Static Public Member Functions**

- static constexpr unsigned short **denorm\_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet\_NaN** () noexcept
- static constexpr unsigned short **round\_error** () noexcept
- static constexpr unsigned short **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.750.1 Detailed Description

numeric\_limits<unsigned short> specialization.

## 6.750.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

**tinyness\_before**

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```

True if tininess is detected before rounding. (see IEC 559)

**traps**

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.751 std::numeric\_limits< wchar\_t > Struct Reference**

```
#include <limits>
```

Inheritance diagram for std::numeric\_limits< wchar\_t >:

**Static Public Member Functions**

- static constexpr wchar\_t **denorm\_min** () noexcept
- static constexpr wchar\_t **epsilon** () noexcept
- static constexpr wchar\_t **infinity** () noexcept
- static constexpr wchar\_t **lowest** () noexcept
- static constexpr wchar\_t **max** () noexcept
- static constexpr wchar\_t **min** () noexcept
- static constexpr wchar\_t **quiet\_NaN** () noexcept
- static constexpr wchar\_t **round\_error** () noexcept
- static constexpr wchar\_t **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int [digits](#)
- static constexpr int **digits10**
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool **has\_infinity**
- static constexpr bool [has\\_infinity](#)
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool **is\_bounded**
- static constexpr bool [is\\_bounded](#)
- static constexpr bool **is\_exact**
- static constexpr bool [is\\_exact](#)
- static constexpr bool **is\_iec559**
- static constexpr bool [is\\_iec559](#)
- static constexpr bool **is\_integer**
- static constexpr bool [is\\_integer](#)
- static constexpr bool **is\_modulo**
- static constexpr bool [is\\_modulo](#)
- static constexpr bool **is\_signed**
- static constexpr bool [is\\_signed](#)
- static constexpr bool **is\_specialized**
- static constexpr bool [is\\_specialized](#)
- static constexpr int **max\_digits10**
- static constexpr int [max\\_digits10](#)
- static constexpr int **max\_exponent**
- static constexpr int [max\\_exponent](#)
- static constexpr int **max\_exponent10**
- static constexpr int [max\\_exponent10](#)
- static constexpr int **min\_exponent**
- static constexpr int [min\\_exponent](#)
- static constexpr int **min\_exponent10**
- static constexpr int [min\\_exponent10](#)
- static constexpr int **radix**
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool **tinyness\_before**
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool **traps**
- static constexpr bool [traps](#)

### 6.751.1 Detailed Description

numeric\_limits<wchar\_t> specialization.



## 6.751.2 Member Data Documentation

### digits

```
int std::__numeric_limits_base::digits [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

### digits10

```
int std::__numeric_limits_base::digits10 [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

### has\_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
```

See `std::float_denorm_style` for more information.

### has\_denorm\_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

### has\_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```

True if the type has a representation for positive infinity.

### has\_quiet\_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

### has\_signaling\_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```

True if the type has a representation for a signaling Not a Number.

### is\_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

### is\_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

### is\_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

**is\_integer**

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`  
True if the type is integer.

**is\_modulo**

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`  
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

**is\_signed**

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`  
True if the type is signed.

**is\_specialized**

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`  
This will be true for all fundamental types (which have specializations), and false for everything else.

**max\_digits10**

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`  
The number of base 10 digits required to ensure that values which differ are always differentiated.

**max\_exponent**

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`  
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

**max\_exponent10**

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`  
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

**min\_exponent**

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`  
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

**min\_exponent10**

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`  
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

**radix**

`int std::__numeric_limits_base::radix [static], [constexpr]`  
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

**round\_style**

`float_round_style` `std::__numeric_limits_base::round_style` [static], [constexpr]

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

**tinyness\_before**

`bool` `std::__numeric_limits_base::tinyness_before` [static], [constexpr]

True if tininess is detected before rounding. (see IEC 559)

**traps**

`bool` `std::__numeric_limits_base::traps` [static], [constexpr]

True if trapping is implemented for this type.

The documentation for this struct was generated from the following file:

- [limits](#)

**6.752 std::num\_punct<\_CharT> Class Template Reference**

```
#include <locale_facets.h>
```

Inheritance diagram for `std::num_punct<_CharT>`:

**Public Types**

- `typedef __num_punct_cache<_CharT> __cache_type`
- `typedef _CharT char_type`
- `typedef basic_string<_CharT> string_type`

### Public Member Functions

- [numpunct](#) ([\\_\\_c\\_locale](#) \_\_cloc, [size\\_t](#) \_\_refs=0)
- [numpunct](#) ([\\_\\_cache\\_type](#) \* \_\_cache, [size\\_t](#) \_\_refs=0)
- [numpunct](#) ([size\\_t](#) \_\_refs=0)
- [char\\_type decimal\\_point](#) () const
- [string\\_type falsename](#) () const
- [string grouping](#) () const
- [char\\_type thousands\\_sep](#) () const
- [string\\_type truename](#) () const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~numpunct](#) ()
- void [\\_M\\_initialize\\_numpunct](#) ([\\_\\_c\\_locale](#) \_\_cloc)
- void [\\_M\\_initialize\\_numpunct](#) ([\\_\\_c\\_locale](#) \_\_cloc)
- void [\\_M\\_initialize\\_numpunct](#) ([\\_\\_c\\_locale](#) \_\_cloc=0)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

### Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc, const char \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \* \_\_s)

### Protected Attributes

- [\\_\\_cache\\_type](#) \* [\\_M\\_data](#)

#### 6.752.1 Detailed Description

```
template<typename _CharT>
class std::numpunct<_CharT >
```

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

### 6.752.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT>
typedef _CharT std::numpunct<_CharT>::char_type
```

Public typedefs.

#### string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::numpunct<_CharT>::string_type
```

Public typedefs.

### 6.752.3 Constructor & Destructor Documentation

#### numpunct() [1/3]

```
template<typename _CharT>
std::numpunct<_CharT>::numpunct (
 size_t __refs = 0) [inline], [explicit]
```

Numpunct constructor.

##### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__refs</code> | Refcount to pass to the base class. |
|---------------------|-------------------------------------|

References [std::locale::facet::facet\(\)](#).

#### numpunct() [2/3]

```
template<typename _CharT>
std::numpunct<_CharT>::numpunct (
 __cache_type * __cache,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

##### Parameters

|                      |                                       |
|----------------------|---------------------------------------|
| <code>__cache</code> | <code>__numpunct_cache</code> object. |
| <code>__refs</code>  | Refcount to pass to the base class.   |

References [std::locale::facet::facet\(\)](#).

#### numpunct() [3/3]

```
template<typename _CharT>
std::numpunct<_CharT>::numpunct (
 __c_locale __cloc,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__refs</code> | RefCount to pass to the base class. |

References [std::locale::facet::facet\(\)](#).

**~numpunct()**

```
template<typename _CharT>
virtual std::numpunct< _CharT >::~~numpunct () [protected], [virtual]
Destructor.
```

**6.752.4 Member Function Documentation****decimal\_point()**

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const [inline]
Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning numpunct<char_type>::do_decimal_point\(\).
```

**Returns**

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

**do\_decimal\_point()**

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const [inline], [protected],
[virtual]
Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.
```

**Returns**

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_falsename()**

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_falsename () const [inline], [protected], [virtual]
Return string representation of bool false.
Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.
```

**Returns**

*string\_type* representing printed form of false.

Referenced by [falsename\(\)](#).

**do\_grouping()**

```
template<typename _CharT>
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

grouping() for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT>
virtual char_type std::num_punct<_CharT>::do_thousands_sep () const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**do\_truename()**

```
template<typename _CharT>
virtual string_type std::num_punct<_CharT>::do_truename () const [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

string\_type representing printed form of true.

Referenced by [truename\(\)](#).

**falsename()**

```
template<typename _CharT>
string_type std::num_punct<_CharT>::falsename () const [inline]
```

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_falsename().

**Returns**

string\_type representing printed form of false.

References [do\\_falsename\(\)](#).

**grouping()**

```
template<typename _CharT>
string std::num_punct< _CharT >::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**thousands\_sep()**

```
template<typename _CharT>
char_type std::num_punct< _CharT >::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

**Returns**

`char_type` representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

**true\_name()**

```
template<typename _CharT>
string_type std::num_punct< _CharT >::true_name () const [inline]
```

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `num_punct<char_type>::do_true_name()`.

**Returns**

`string_type` representing printed form of true.

References [do\\_true\\_name\(\)](#).

**6.752.5 Member Data Documentation****id**

```
template<typename _CharT>
locale::id std::num_punct< _CharT >::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)



## 6.753 std::numpunct\_byname<\_CharT> Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::numpunct\_byname<\_CharT>:



### Public Types

- typedef `__numpunct_cache<_CharT>` **`__cache_type`**
- typedef `_CharT` **`char_type`**
- typedef `basic_string<_CharT>` **`string_type`**

### Public Member Functions

- **`numpunct_byname`** (`const char *__s`, `size_t __refs=0`)
- **`numpunct_byname`** (`const string &__s`, `size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

### Static Public Attributes

- static `locale::id` `id`

### Protected Member Functions

- void **`_M_initialize_numpunct`** (`__c_locale __cloc`)
- void **`_M_initialize_numpunct`** (`__c_locale __cloc`)
- void **`_M_initialize_numpunct`** (`__c_locale __cloc=0`)
- virtual `char_type do_decimal_point` () const
- virtual `string_type do_falsename` () const
- virtual `string do_grouping` () const

- virtual char\_type [do\\_thousands\\_sep](#) () const
- virtual string\_type [do\\_truename](#) () const

### Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_cache\_type \* [\\_M\\_data](#)

### 6.753.1 Detailed Description

**template<typename \_CharT>**  
**class std::numpunct\_byname< \_CharT >**

class numpunct\_byname [22.2.3.2].

### 6.753.2 Member Function Documentation

#### decimal\_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning [numpunct<char\\_type>::do\\_decimal\\_point\(\)](#).

#### Returns

*char\_type* representing a decimal point.

References [do\\_decimal\\_point\(\)](#).

#### do\_decimal\_point()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a char\_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a decimal point.

Referenced by [decimal\\_point\(\)](#).

**do\_falsename()**

```
template<typename _CharT>
virtual string_type std::num_punct<_CharT>::do_falsename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

`string_type` representing printed form of false.

Referenced by [falsename\(\)](#).

**do\_grouping()**

```
template<typename _CharT>
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

**See also**

[grouping\(\)](#) for details.

**Returns**

String representing grouping specification.

Referenced by [grouping\(\)](#).

**do\_thousands\_sep()**

```
template<typename _CharT>
virtual char_type std::num_punct<_CharT>::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Referenced by [thousands\\_sep\(\)](#).

**do\_truename()**

```
template<typename _CharT>
virtual string_type std::num_punct<_CharT>::do_truename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

`string_type` representing printed form of true.

Referenced by [truename\(\)](#).

**falsename()**

```
template<typename _CharT>
```

```
string_type std::num_punct< _CharT >::false_name () const [inline], [inherited]
```

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_false\_name().

**Returns**

string\_type representing printed form of false.

References [do\\_false\\_name\(\)](#).

**grouping()**

```
template<typename _CharT>
```

```
string std::num_punct< _CharT >::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

**Returns**

string representing grouping specification.

References [do\\_grouping\(\)](#).

**thousands\_sep()**

```
template<typename _CharT>
```

```
char_type std::num_punct< _CharT >::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

**Returns**

char\_type representing a thousands separator.

References [do\\_thousands\\_sep\(\)](#).

**true\_name()**

```
template<typename _CharT>
```

```
string_type std::num_punct< _CharT >::true_name () const [inline], [inherited]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling num\_punct<char\_type>::do\_true\_name().

**Returns**

string\_type representing printed form of true.

References [do\\_true\\_name\(\)](#).

### 6.753.3 Member Data Documentation

#### id

template<typename \_CharT>

[locale::id](#) [std::numpunct](#)< \_CharT >::id [static], [inherited]

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 6.754 \_\_gnu\_parallel::omp\_loop\_static\_tag Struct Reference

#include <tags.h>

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_static\_tag:



### Public Member Functions

- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

### 6.754.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

### 6.754.2 Member Function Documentation

#### [\\_\\_get\\_num\\_threads](#)()

[\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::parallel\\_tag::\\_\\_get\\_num\\_threads](#) () [inline], [inherited]

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)(), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort](#)()

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**6.755 \_\_gnu\_parallel::omp\_loop\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:

**Public Member Functions**

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \(\)](#)
- void [set\\_num\\_threads \(\\_ThreadIndex \\_\\_num\\_threads\)](#)

**6.755.1 Detailed Description**

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

**6.755.2 Member Function Documentation****\_\_get\_num\_threads()**

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**6.756 std::once\_flag Struct Reference**

```
#include <mutex>
```

**Public Member Functions**

- [once\\_flag](#) (const once\_flag &)=delete
- once\_flag & [operator=](#) (const once\_flag &)=delete

**Friends**

- `template<typename _Callable, typename... _Args>`  
void [call\\_once](#) (once\_flag &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)

**6.756.1 Detailed Description**

Flag type used by std::call\_once.

**6.756.2 Constructor & Destructor Documentation****once\_flag()**

```
std::once_flag::once_flag (
 const once_flag &) [delete]
```

Deleted copy constructor.

**6.756.3 Member Function Documentation****operator=()**

```
once_flag & std::once_flag::operator= (
 const once_flag &) [delete]
```

Deleted assignment operator.

**6.756.4 Friends And Related Symbol Documentation****call\_once**

```
template<typename _Callable, typename... _Args>
void call_once (
 once_flag & __once,
 _Callable && __f,
 _Args &&... __args) [friend]
```

Invoke a callable and synchronize with other calls using the same flag.  
References [std::\\_\\_invoke\(\)](#), and [std::forward\(\)](#).  
The documentation for this struct was generated from the following file:

- [mutex](#)

## 6.757 std::experimental::fundamentals\_v1::optional<\_Tp> Class Template Reference

```
#include <optional>
```

### Public Types

- using [value\\_type](#)

### Public Member Functions

- template<typename \_Up = \_Tp, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< optional< \_Tp >, [decay\\_t](#)< \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not\_< is\_convertible< \_Up &&, \_Tp > >::value, bool > = false>  
constexpr **optional** ( \_Up &&\_\_t)
- template<typename \_Up = \_Tp, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< optional< \_Tp >, [decay\\_t](#)< \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp > >::value, bool > = true>  
constexpr **optional** ( \_Up &&\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, \_\_not\_< is\_convertible< const \_Up &, \_Tp > >, \_\_not\_< \_\_converts\_from\_optional< \_Tp, \_Up > >::value, bool > = false>  
constexpr **optional** (const optional< \_Up > &\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, is\_↵convertible< const \_Up &, \_Tp > >, \_\_not\_< \_\_converts\_from\_optional< \_Tp, \_Up > >::value, bool > = true>  
constexpr **optional** (const optional< \_Up > &\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, \_\_not\_< is\_↵convertible< \_Up &&, \_Tp > >, \_\_not\_< \_\_converts\_from\_optional< \_Tp, \_Up > >::value, bool > = false>  
constexpr **optional** (optional< \_Up > &&\_\_t)
- template<typename \_Up, [enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp > >, \_\_not\_< \_\_converts\_from\_optional< \_Tp, \_Up > >::value, bool > = true>  
constexpr **optional** (optional< \_Up > &&\_\_t)
- template<typename... \_Args>  
[enable\\_if\\_t](#)< [is\\_constructible](#)< \_Tp, \_Args &&... >::value > **emplace** ( \_Args &&... \_\_args)
- template<typename \_Up, typename... \_Args>  
[enable\\_if\\_t](#)< [is\\_constructible](#)< \_Tp, [initializer\\_list](#)< \_Up > &, \_Args &&... >::value > **emplace** ([initializer\\_list](#)< \_Up > \_\_il, \_Args &&... \_\_args)
- constexpr **operator bool** () const noexcept
- constexpr \_Tp & **operator\*** () &
- constexpr \_Tp && **operator\*** () &&
- constexpr const \_Tp & **operator\*** () const &
- constexpr const \_Tp && **operator\*** () const &&
- \_Tp \* **operator->** ()
- constexpr const \_Tp \* **operator->** () const
- template<typename \_Up = \_Tp>  
[enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< optional< \_Tp >, [decay\\_t](#)< \_Up > >, [is\\_constructible](#)< \_Tp, \_Up >, \_\_not\_< \_\_and\_< [is\\_scalar](#)< \_Tp >, is\_same< \_Tp, [decay\\_t](#)< \_Up > > >, [is\\_assignable](#)< \_Tp &, \_Up > >::value, optional & > **operator=** ( \_Up &&\_\_u)
- template<typename \_Up>  
[enable\\_if\\_t](#)< \_\_and\_< \_\_not\_< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, const \_Up & >, [is\\_assignable](#)< \_Tp &, \_Up >, \_\_not\_< \_\_converts\_from\_optional< \_Tp, \_Up > >, \_\_not\_< \_\_assigns\_↵\_from\_optional< \_Tp, \_Up > > >::value, optional & > **operator=** (const optional< \_Up > &\_\_u)



- optional & **operator=** ([nullopt\\_t](#)) noexcept
- template<typename \_Up>  
[enable\\_if\\_t](#)< \_\_and< \_\_not< is\_same< \_Tp, \_Up > >, [is\\_constructible](#)< \_Tp, \_Up >, [is\\_assignable](#)< \_Tp &, \_Up >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up > >, \_\_not< \_\_assigns\_from\_optional< \_Tp, \_Up > > >::value, optional & > **operator=** (optional< \_Up > &&\_\_u)
- void **swap** (optional &\_\_other) noexcept([is\\_nothrow\\_move\\_constructible](#)< \_Tp >()) &&\_\_is\_nothrow\_↵  
swappable< \_Tp >::value)
- constexpr \_Tp & **value** () &
- constexpr \_Tp && **value** () &&
- constexpr const \_Tp & **value** () const &
- constexpr const \_Tp && **value** () const &&
- template<typename \_Up>  
\_Tp **value\_or** (\_Up &&\_\_u) &&
- template<typename \_Up>  
constexpr \_Tp **value\_or** (\_Up &&\_\_u) const &

### 6.757.1 Detailed Description

template<typename \_Tp>  
class std::experimental::fundamentals\_v1::optional< \_Tp >

Class template for optional values.

The documentation for this class was generated from the following file:

- [experimental/optional](#)

## 6.758 std::ostream\_iterator< \_Tp, \_CharT, \_Traits > Class Template Reference

```
#include <stream_iterator.h>
```

Inheritance diagram for std::ostream\_iterator< \_Tp, \_CharT, \_Traits >:



### Public Types

- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)

- typedef void [value\\_type](#)
- using [difference\\_type](#)
- typedef [\\_CharT](#) [char\\_type](#)
- typedef [\\_Traits](#) [traits\\_type](#)
- typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

## Public Member Functions

- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj) noexcept
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c) noexcept
- [ostream\\_iterator](#) & **operator\*** () noexcept
- [ostream\\_iterator](#) & **operator++** () noexcept
- [ostream\\_iterator](#) & **operator++** (int) noexcept
- [ostream\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_value)
- [ostream\\_iterator](#) & **operator=** (const [ostream\\_iterator](#) &)=default

### 6.758.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

### Template Parameters

|                      |                                        |
|----------------------|----------------------------------------|
| <code>_Tp</code>     | The type to write to the ostream.      |
| <code>_CharT</code>  | The ostream <code>char_type</code> .   |
| <code>_Traits</code> | The ostream <code>char_traits</code> . |

### 6.758.2 Member Typedef Documentation

#### `char_type`

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type
Public typedef.
```

#### `difference_type`

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
using std::ostream_iterator< _Tp, _CharT, _Traits >::difference_type
Public typedef.
```

#### `iterator_category`

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
iterator_category [inherited]
One of the tag types.
```

**ostream\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_↵
type
Public typedef.
```

**pointer**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
This type represents a pointer-to-value_type.
```

**reference**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
This type represents a reference-to-value_type.
```

**traits\_type**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
Public typedef.
```

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
The type "pointed to" by the iterator.
```

**6.758.3 Constructor & Destructor Documentation****ostream\_iterator() [1/3]**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 ostream_type & __s) [inline], [noexcept]
```

Construct from an ostream.

References [std::\\_\\_addressof\(\)](#).

Referenced by [ostream\\_iterator\(\)](#).

**ostream\_iterator() [2/3]**

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 ostream_type & __s,
 const _CharT * __c) [inline], [noexcept]
```

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

**Parameters**

|                                          |                                   |
|------------------------------------------|-----------------------------------|
| <a href="#">_↵</a><br><a href="#">_s</a> | Underlying ostream to write to.   |
| <a href="#">_↵</a><br><a href="#">_c</a> | CharT delimiter string to insert. |

References [std::\\_\\_addressof\(\)](#).

### ostream\_iterator() [3/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 const ostream_iterator< _Tp, _CharT, _Traits > & __obj) [inline], [noexcept]
```

Copy constructor.

References [ostream\\_iterator\(\)](#).

## 6.758.4 Member Function Documentation

### operator=()

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator & std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
 const _Tp & __value) [inline]
```

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

## 6.759 std::experimental::fundamentals\_v2::ostream\_joiner< \_DelimT, \_CharT, \_Traits > Class Template Reference

```
#include <iterator>
```

### Public Types

- typedef `_CharT` **char\_type**
- typedef void **difference\_type**
- typedef [output\\_iterator\\_tag](#) **iterator\_category**
- typedef [basic\\_ostream](#)< `_CharT`, `_Traits` > **ostream\_type**
- typedef void **pointer**
- typedef void **reference**
- typedef `_Traits` **traits\_type**
- typedef void **value\_type**

### Public Member Functions

- **ostream\_joiner** (ostream\_type &\_\_os, \_DelimT &&\_\_delimiter) noexcept(is\_nothrow\_move\_constructible\_v< \_DelimT >)
- **ostream\_joiner** (ostream\_type &\_\_os, const \_DelimT &\_\_delimiter) noexcept(is\_nothrow\_copy\_constructible\_v< \_DelimT >)
- ostream\_joiner & **operator\*** () noexcept
- ostream\_joiner & **operator++** () noexcept
- ostream\_joiner & **operator++** (int) noexcept
- template<typename \_Tp>  
ostream\_joiner & **operator=** (const \_Tp &\_\_value)

### 6.759.1 Detailed Description

```
template<typename _DelimT, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >
```

Output iterator that inserts a delimiter between elements.

The documentation for this class was generated from the following file:

- [experimental/iterator](#)

## 6.760 std::ostreambuf\_iterator< \_CharT, \_Traits > Class Template Reference

```
#include <ostreambuf_iterator.h>
```

Inheritance diagram for std::ostreambuf\_iterator< \_CharT, \_Traits >:



### Public Types

- typedef [output\\_iterator\\_tag](#) `iterator_category`
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)
- using [difference\\_type](#)
- typedef `_CharT` [char\\_type](#)
- typedef `_Traits` [traits\\_type](#)
- typedef [basic\\_ostreambuf\\_iterator< \\_CharT, \\_Traits >](#) [ostreambuf\\_type](#)
- typedef [basic\\_ostream\\_iterator< \\_CharT, \\_Traits >](#) [ostream\\_type](#)

### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([ostreambuf\\_type](#) \*\_\_s) noexcept
- [ostreambuf\\_iterator](#) & [M\\_put](#) (const `_CharT` \*\_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()

- ostreambuf\_iterator & [operator++](#) ()
- ostreambuf\_iterator & [operator++](#) (int)
- ostreambuf\_iterator & [operator=](#) (\_CharT \_\_c)

## Friends

- `template<typename _CharT2>  
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type >::__type copy  
(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`

## 6.760.1 Detailed Description

**template<typename \_CharT, typename \_Traits>**  
**class std::ostreambuf\_iterator< \_CharT, \_Traits >**

Provides output iterator semantics for streambufs.

## 6.760.2 Member Typedef Documentation

### char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type
Public typedefs.
```

### difference\_type

```
template<typename _CharT, typename _Traits>
using std::ostreambuf_iterator< _CharT, _Traits >::difference_type
Public typedefs.
```

### iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
iterator_category [inherited]
One of the tag types.
```

### ostream\_type

```
template<typename _CharT, typename _Traits>
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
Public typedefs.
```

### pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
This type represents a pointer-to-value_type.
```

### reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
This type represents a reference-to-value_type.
```

**streambuf\_type**

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf←
_type
Public typedefs.
```

**traits\_type**

```
template<typename _CharT, typename _Traits>
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
Public typedefs.
```

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
The type "pointed to" by the iterator.
```

**6.760.3 Constructor & Destructor Documentation****ostreambuf\_iterator() [1/2]**

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
 ostream_type & __s) [inline], [noexcept]
```

Construct output iterator from ostream.

**ostreambuf\_iterator() [2/2]**

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
 streambuf_type * __s) [inline], [noexcept]
```

Construct output iterator from streambuf.

**6.760.4 Member Function Documentation****failed()**

```
template<typename _CharT, typename _Traits>
bool std::ostreambuf_iterator< _CharT, _Traits >::failed () const [inline], [nodiscard], [noexcept]
```

Return true if previous operator=() failed.

Referenced by [std::basic\\_ostream<\\_CharT, \\_Traits>::operator<<\(\)](#).

**operator\*()**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator* () [inline], [nodiscard]
Return *this.
```

**operator++() [1/2]**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
Return *this.
```

**operator++()** [2/2]

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
 int) [inline]
```

Return \*this.

**operator=()**

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator= (
 _CharT __c) [inline]
```

Write character to streambuf. Calls streambuf.sputc().

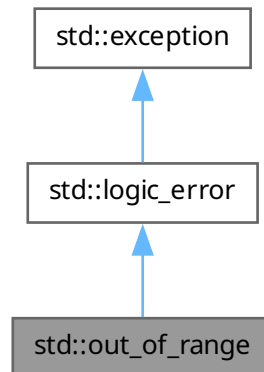
The documentation for this class was generated from the following files:

- [iosfwd.h](#)
- [streambuf\\_iterator.h](#)

**6.761 std::out\_of\_range Class Reference**

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::out\_of\_range:

**Public Member Functions**

- **out\_of\_range** (const char \*)
- **out\_of\_range** (const out\_of\_range &)=default
- **out\_of\_range** (const [string](#) &\_\_arg)
- **out\_of\_range** (out\_of\_range &&)=default
- out\_of\_range & **operator=** (const out\_of\_range &)=default
- out\_of\_range & **operator=** (out\_of\_range &&)=default
- virtual const char \* **what** () const noexcept



### 6.761.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in `basic_string`).

### 6.761.2 Member Function Documentation

#### what()

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept\\_except.h](#)

## 6.762 std::out\_ptr\_t< \_Smart, \_Pointer, \_Args > Class Template Reference

```
#include <>>
```

### Public Member Functions

- **out\_ptr\_t** (`_Smart &__smart, _Args... __args`)
- **out\_ptr\_t** (`const out_ptr_t &)=delete`
- **operator \_Pointer \*** (`() const noexcept`)
- **operator void \*\*** (`() const noexcept`)

### Friends

- `template<typename, typename, typename...>`  
`class inout_ptr_t`

### 6.762.1 Detailed Description

**template<typename \_Smart, typename \_Pointer, typename... \_Args>**

**class std::out\_ptr\_t< \_Smart, \_Pointer, \_Args >**

Smart pointer adaptor for functions taking an output pointer parameter.

#### Template Parameters

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <code>_Smart</code>   | The type of pointer to adapt.                         |
| <code>_Pointer</code> | The type of pointer to convert to.                    |
| <code>_Args...</code> | Argument types used when resetting the smart pointer. |

Since

C++23

The documentation for this class was generated from the following file:

- [out\\_ptr.h](#)

## 6.763 std::output\_iterator\_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

### 6.763.1 Detailed Description

Marking output iterators.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 6.764 \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > Class Template Reference

```
#include <ov_tree_map_.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >:



### Classes

- class [cond\\_dtor](#)

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `ov_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `traits_type::node_update` **node\_update**
- typedef `const_pointer` **point\_const\_iterator**
- typedef `pointer` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

### Public Member Functions

- **ov\_tree\_map** (const Cmp\_Fn &)
- **ov\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **ov\_tree\_map** (const ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key\_const\_reference)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference r\_key)
- point\_const\_iterator **find** (key\_const\_reference r\_key) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const

- `std::pair< point_iterator, bool >` **insert** (const\_reference r\_value)
- void **join** (ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference r\_key)
- point\_const\_iterator **lower\_bound** (key\_const\_reference r\_key) const
- size\_type **max\_size** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_end** ()
- node\_const\_iterator **node\_end** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **split** (key\_const\_reference, ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference r\_key)
- point\_const\_iterator **upper\_bound** (key\_const\_reference r\_key) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### 6.764.1 Detailed Description

**template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_↵ Alloc>**

**class \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >**

Ordered-vector tree associative-container.

### 6.764.2 Member Function Documentation

#### **node\_begin()** [1/2]

**template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_↵ Alloc>**

**ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_iterator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_begin ()** [inline]

Returns a node\_iterator corresponding to the node at the root of the tree.

#### **node\_begin()** [2/2]

**template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_↵ Alloc>**

**ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_const\_iterator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_begin () const** [inline]

Returns a const node\_iterator corresponding to the node at the root of the tree.

#### **node\_end()** [1/2]

**template<typename Key, typename Mapped, typename Cmp\_Fn, typename Node\_And\_It\_Traits, typename \_↵ Alloc>**

**ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_iterator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::node\_end ()** [inline]

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

## node\_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_r
```

```
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

## 6.765 `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >`:



## Public Types

- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::const\_pointer **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Metadata\_Type **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::const\_pointer **reference**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::const\_pointer **value\_type**

## Public Member Functions

- **ov\_tree\_node\_const\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- this\_type [get\\_l\\_child](#) () const

- metadata\_const\_reference **get\_metadata** () const
- this\_type **get\_r\_child** () const
- bool **operator!=** (const this\_type &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const this\_type &other) const

#### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

#### Protected Types

- typedef [rebind\\_traits](#)< \_Alloc, Metadata\_Type >::const\_pointer **const\_metadata\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::const\_pointer **const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::pointer **pointer**
- typedef ov\_tree\_node\_const\_it\_< Value\_Type, Metadata\_Type, \_Alloc > **this\_type**

#### Static Protected Member Functions

- template<typename Ptr>  
static Ptr **mid\_pointer** (Ptr p\_begin, Ptr p\_end)

#### 6.765.1 Detailed Description

template<typename Value\_Type, typename Metadata\_Type, typename \_Alloc>  
class **\_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it\_** < Value\_Type, Metadata\_Type, \_Alloc >

Const node reference.

#### 6.765.2 Member Function Documentation

##### **get\_l\_child()**

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_l_↵
l_child () const [inline]
```

Returns the node iterator associated with the left node.

##### **get\_r\_child()**

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↵
r_child () const [inline]
```

Returns the node iterator associated with the right node.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

## 6.766 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::pointer **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Metadata\_Type **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, typename remove\_const< Value\_Type >::type >::pointer **reference**
- typedef [rebind\\_traits](#)< \_Alloc, Value\_Type >::pointer **value\_type**

### Public Member Functions

- **ov\_tree\_node\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_↵ metadata\_pointer p\_metadata=0)
- **ov\_tree\_node\_it\_** [get\\_l\\_child](#) () const
- metadata\_const\_reference **get\_metadata** () const
- **ov\_tree\_node\_it\_** [get\\_r\\_child](#) () const
- bool **operator!=** (const this\_type &other) const
- reference [operator\\*](#) () const
- bool **operator==** (const this\_type &other) const

### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Static Protected Member Functions

- `template<typename Ptr>`  
`static Ptr mid_pointer (Ptr p_begin, Ptr p_end)`

### 6.766.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >
```

Node reference.

### 6.766.2 Member Function Documentation

#### `get_l_child()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_l_child () const [inline]
```

Returns the node reference associated with the left node.

#### `get_r_child()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_r_child () const [inline]
```

Returns the node reference associated with the right node.

#### `operator*()`

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ()
const [inline]
```

Access.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

## 6.767 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```



Inheritance diagram for \_\_gnu\_pbds::ov\_tree\_tag:



#### 6.767.1 Detailed Description

Ordered-vector tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.768 std::overflow\_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- **overflow\_error** (const char \*)
- **overflow\_error** (const overflow\_error &)=default
- **overflow\_error** (const [string](#) &\_\_arg)
- **overflow\_error** (overflow\_error &&)=default
- overflow\_error & **operator=** (const overflow\_error &)=default
- overflow\_error & **operator=** (overflow\_error &&)=default
- virtual const char \* [what](#) () const noexcept

#### 6.768.1 Detailed Description

Thrown to indicate arithmetic overflow.

#### 6.768.2 Member Function Documentation

##### **what()**

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept\\_except.h](#)

## 6.769 std::owner\_less< \_Tp > Struct Template Reference

### 6.769.1 Detailed Description

```
template<typename _Tp = void>
struct std::owner_less< _Tp >
```

Primary template owner\_less.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 6.770 std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 6.770.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
```

Partial specialization of owner\_less for shared\_ptr.

### 6.770.2 Member Typedef Documentation

#### first\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

## 6.771 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept
- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const noexcept
- bool **operator()** (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept

#### 6.771.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

#### 6.771.2 Member Typedef Documentation

##### `first_argument_type`

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

##### `result_type`

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type
```

##### `second_argument_type`

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:
```

- [bits/shared\\_ptr.h](#)

## 6.772 `std::owner_less< void >` Struct Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept
- bool **operator()** (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const noexcept
- bool **operator()** (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept

### 6.772.1 Detailed Description

Void specialization of owner\_less compares either shared\_ptr or weak\_ptr.

### 6.772.2 Member Typedef Documentation

#### first\_argument\_type

typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]  
first\_argument\_type is the type of the first argument

#### result\_type

typedef bool std::binary\_function< \_Tp, \_Tp, bool >::result\_type [inherited]  
result\_type is the return type

#### second\_argument\_type

typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]  
second\_argument\_type is the type of the second argument

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 6.773 std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 6.773.1 Detailed Description

template<typename \_Tp>

struct std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >

Partial specialization of owner\_less for weak\_ptr.

### 6.773.2 Member Typedef Documentation

#### first\_argument\_type

typedef \_Tp std::binary\_function< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]  
first\_argument\_type is the type of the first argument

**result\_type**

typedef bool [std::binary\\_function](#)< \_Tp, \_Tp, bool >::result\_type [inherited]  
result\_type is the return type

**second\_argument\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

**6.774 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference**

#include <shared\_ptr.h>

**Public Types**

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

**Public Member Functions**

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

**6.774.1 Detailed Description**

**template<typename \_Tp>**  
**struct std::owner\_less< weak\_ptr< \_Tp > >**

Partial specialization of owner\_less for weak\_ptr.

**6.774.2 Member Typedef Documentation****first\_argument\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::first\_argument\_type [inherited]  
first\_argument\_type is the type of the first argument

**result\_type**

typedef bool [std::binary\\_function](#)< \_Tp, \_Tp, bool >::result\_type [inherited]  
result\_type is the return type

**second\_argument\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, bool >::second\_argument\_type [inherited]  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 6.775 std::packaged\_task< \_Res(\_ArgTypes...)> Class Template Reference

```
#include <future>
```

### Public Member Functions

- template<typename \_Fn, typename = \_\_not\_same<\_Fn>>>  
**packaged\_task** (\_Fn &&\_\_fn)
- template<typename \_Fn, typename \_Alloc, typename = \_\_not\_same<\_Fn>>>  
**packaged\_task** (allocator\_arg\_t, const \_Alloc &\_\_a, \_Fn &&\_\_fn)
- **packaged\_task** (const packaged\_task &)=delete
- **packaged\_task** (packaged\_task &&\_\_other) noexcept
- [future](#)< \_Res > **get\_future** ()
- void **make\_ready\_at\_thread\_exit** (\_ArgTypes... \_\_args)
- void **operator()** (\_ArgTypes... \_\_args)
- packaged\_task & **operator=** (const packaged\_task &)=delete
- packaged\_task & **operator=** (packaged\_task &&\_\_other) noexcept
- void **reset** ()
- void **swap** (packaged\_task &\_\_other) noexcept
- bool **valid** () const noexcept

### 6.775.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::packaged_task< _Res(_ArgTypes...)>
```

packaged\_task

The documentation for this class was generated from the following file:

- [future](#)

## 6.776 \_\_gnu\_cxx::pair< \_T1, \_T2 > Struct Template Reference

```
#include <utility>
```

### Public Types

- typedef \_T1 [first\\_type](#)
- typedef \_T2 [second\\_type](#)

### Public Member Functions

- constexpr [pair](#) (const pair &)=default
- constexpr [pair](#) (pair &&)=default
- template<typename... \_Args1, typename... \_Args2>  
constexpr [pair](#) (piecewise\_construct\_t, tuple< \_Args1... >, tuple< \_Args2... >)
- template<typename \_U1, typename \_U2>  
constexpr **explicit** (!\_S\_convertible< \_U1 &, \_U2 & >()) [pair](#)([pair](#)< \_U1
- template<typename \_U1, typename \_U2>  
constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible< \_U1 &, \_U2 & >()) [pair](#)([pair](#)< \_U1
- template<typename \_U1 = \_T1, typename \_U2 = \_T2>  
constexpr **explicit** (!\_S\_convertible< \_U1, \_U2 >()) [pair](#)(\_U1 &&
- template<typename \_U1 = \_T1, typename \_U2 = \_T2>  
constexpr **explicit** (!\_S\_convertible< \_U1, \_U2 >()) [pair](#)(\_U1 &&\_\_x

- `template<typename _U1, typename _U2>`  
`constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1`
- `template<typename _U1, typename _U2>`  
`constexpr _U2 &&constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1`
- `constexpr explicit (!_S_convertible< const _T1 &, const _T2 & >()) pair(const type_identity_t< _T1 > &__x`
- `template<typename _U1, typename _U2>`  
`constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1`
- `template<typename _U1, typename _U2>`  
`constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1`
- `template<typename _U1, typename _U2>`  
`constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1`
- `template<typename _U1, typename _U2>`  
`constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1`
- `template<__eligible_pair_like< pair > _UPair>`  
`constexpr explicit (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&)=delete`
- `template<__eligible_pair_like< pair > _UPair>`  
`constexpr _U2 &&constexpr explicit (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&__p)`
- `constexpr explicit (__not< __and< __is_implicitly_default_constructible< _T1 >, __is_implicitly_default_↵`  
`constructible< _T2 > > >()) pair() noexcept(is_nothrow_default_constructible_v< _T1 > &&is_nothrow_↵`  
`default_constructible_v< _T2 >)`
- `constexpr _U2 &&constexpr _U2 &__p noexcept (_S_nothrow_constructible< _U1 &, _U2 & >())`
- `constexpr _U2 &&constexpr _U2 &&__p noexcept (_S_nothrow_constructible< _U1, _U2 >())`
- `constexpr _U2 &&__y noexcept (_S_nothrow_constructible< _U1, _U2 >())`
- `constexpr const _T2 &__y noexcept (_S_nothrow_constructible< const _T1 &, const _T2 & >())`
- `constexpr _U2 &&constexpr _U2 &__p noexcept (_S_nothrow_constructible< const _U1 &, const _U2 & >())`
- `constexpr _U2 &&constexpr _U2 &&__p noexcept (_S_nothrow_constructible< const _U1, const _U2 >())`
- `template<__eligible_pair_like< pair > _UPair>`  
`constexpr pair & operator= (_UPair &&__p)`
- `template<__eligible_pair_like< pair > _UPair>`  
`constexpr const pair & operator= (_UPair &&__p) const`
- `pair & operator= (const pair &)=delete`
- `constexpr const pair & operator= (const pair &__p) const`
- `constexpr pair & operator= (const pair &__p) noexcept(_S_nothrow_assignable< const _T1 &, const _T2 & >())`
- `template<typename _U1, typename _U2>`  
`requires (_S_const_assignable<const _U1&, const _U2&>())`  
`constexpr const pair & operator= (const pair< _U1, _U2 > &__p) const`
- `template<typename _U1, typename _U2>`  
`requires (_S_assignable<const _U1&, const _U2&>())`  
`constexpr pair & operator= (const pair< _U1, _U2 > &__p) noexcept(_S_nothrow_assignable< const _U1 &,`  
`const _U2 & >())`
- `constexpr const pair & operator= (pair &&__p) const`
- `constexpr pair & operator= (pair &&__p) noexcept(_S_nothrow_assignable< _T1, _T2 >())`
- `template<typename _U1, typename _U2>`  
`requires (_S_const_assignable<_U1, _U2>())`  
`constexpr const pair & operator= (pair< _U1, _U2 > &&__p) const`
- `template<typename _U1, typename _U2>`  
`requires (_S_assignable<_U1, _U2>())`  
`constexpr pair & operator= (pair< _U1, _U2 > &&__p) noexcept(_S_nothrow_assignable< _U1, _U2 >())`
- `constexpr void swap (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _T1 >, _↵`  
`_is_nothrow_swappable< const _T2 > >)`
- `constexpr void swap (pair &__p) noexcept(__and< __is_nothrow_swappable< _T1 >, __is_nothrow_↵`  
`swappable< _T2 > >::value)`



## Public Attributes

- `_T1` [first](#)
- `_T2` [second](#)

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _T1, typename _T2>`  
`pair (_T1, _T2) -> pair<_T1, _T2>`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if<__and<__is_swappable<_T1>, __is_swappable<_T2>>::value>::type swap (pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t<__detail::__synth3way_t<_T1, _U1>, __detail::__synth3way_t<_T2, _U2>> operator<=> (const pair<_T1, _T2> &__x, const pair<_U1, _U2> &__y)`

### 6.776.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_cxx::pair<_T1, _T2>
```

Struct holding two objects (or references) of arbitrary type.

## Template Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>_T1</code> | Type of the <code>first</code> member.  |
| <code>_T2</code> | Type of the <code>second</code> member. |

<https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

### 6.776.2 Member Typedef Documentation

#### `first_type`

```
template<typename _T1, typename _T2>
typedef _T1 std::pair<_T1, _T2>::first_type
```

The type of the `first` member.

#### `second_type`

```
template<typename _T1, typename _T2>
typedef _T2 std::pair<_T1, _T2>::second_type
```

The type of the `second` member.

### 6.776.3 Constructor & Destructor Documentation

#### `pair()` [1/2]

```
template<typename _T1, typename _T2>
std::pair<_T1, _T2>::pair (
 const pair<_T1, _T2> &) [constexpr], [default]
```

Copy constructor.

**pair()** [2/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
 pair< _T1, _T2 > &&) [constexpr], [default]
```

Move constructor.

**6.776.4 Member Function Documentation****explicit()** [1/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1 &, _U2 & > ()) [constexpr]
```

Converting constructor from a non-const pair<U1, U2> lvalue.

**explicit()** [2/7]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2>
std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) && [constexpr]
```

Constructor accepting two values of arbitrary types.

**explicit()** [3/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) [constexpr]
```

Converting constructor from a non-const pair<U1, U2> rvalue.

**explicit()** [4/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _T1 &, const _T2 & > ()) const & [constexpr]
```

Constructor accepting lvalues of first\_type and second\_type.

**explicit()** [5/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _U1 &, const _U2 & > ()) const [constexpr]
```

Converting constructor from a const pair<U1, U2> lvalue.

**explicit()** [6/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _U1, const _U2 > ()) const [constexpr]
```

Converting constructor from a const pair<U1, U2> rvalue.

**explicit()** [7/7]

```
template<typename _T1, typename _T2>
std::pair<_T1, _T2>::explicit (
 __not_< __and_< __is_implicitly_default_constructible<_T1>, __is_implicitly_↵
default_constructible<_T2> > > > ()) [inline], [constexpr], [noexcept]
Default constructor.
```

**operator=()** [1/8]

```
template<typename _T1, typename _T2>
const pair & std::pair<_T1, _T2>::operator= (
 const pair<_T1, _T2> & __p) const [inline], [constexpr]
Copy assignment operator (const).
```

**operator=()** [2/8]

```
template<typename _T1, typename _T2>
pair & std::pair<_T1, _T2>::operator= (
 const pair<_T1, _T2> & __p) [inline], [constexpr], [noexcept]
Copy assignment operator.
```

**operator=()** [3/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<const _U1&, const _U2&>())
const pair & std::pair<_T1, _T2>::operator= (
 const pair<_U1, _U2> & __p) const [inline], [constexpr]
Converting assignment from a const pair<U1, U2> lvalue.
```

**operator=()** [4/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<const _U1&, const _U2&>())
pair & std::pair<_T1, _T2>::operator= (
 const pair<_U1, _U2> & __p) [inline], [constexpr], [noexcept]
Converting assignment from a const pair<U1, U2> lvalue.
```

**operator=()** [5/8]

```
template<typename _T1, typename _T2>
const pair & std::pair<_T1, _T2>::operator= (
 pair<_T1, _T2> && __p) const [inline], [constexpr]
Move assignment operator (const).
```

**operator=()** [6/8]

```
template<typename _T1, typename _T2>
pair & std::pair<_T1, _T2>::operator= (
 pair<_T1, _T2> && __p) [inline], [constexpr], [noexcept]
Move assignment operator.
```

**operator=()** [7/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<_U1, _U2>())
const pair & std::pair< _T1, _T2 >::operator= (
 pair< _U1, _U2 > && __p) const [inline], [constexpr]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

**operator=()** [8/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<_U1, _U2>())
pair & std::pair< _T1, _T2 >::operator= (
 pair< _U1, _U2 > && __p) [inline], [constexpr], [noexcept]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

**swap()**

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
 pair< _T1, _T2 > & __p) [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

**6.776.5 Member Data Documentation****first**

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
The first member.
```

**second**

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
The second member.
```

The documentation for this struct was generated from the following files:

- [bits/stl\\_iterator.h](#)
- [stl\\_pair.h](#)
- [tuple](#)

**6.777 std::pair< \_T1, \_T2 > Struct Template Reference**

```
#include <utility>
```

**Public Types**

- typedef \_T1 [first\\_type](#)
- typedef \_T2 [second\\_type](#)

## Public Member Functions

- constexpr [pair](#) (const pair &)=default
- constexpr [pair](#) (pair &&)=default
- template<typename... \_Args1, typename... \_Args2>  
constexpr [pair](#) ([piecewise\\_construct\\_t](#), [tuple](#)<\_Args1... >, [tuple](#)<\_Args2... >)
- template<typename \_U1, typename \_U2>  
constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) [pair](#)([pair](#)<\_U1
- template<typename \_U1, typename \_U2>  
constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible<\_U1 &, \_U2 &>()) [pair](#)([pair](#)<\_U1
- template<typename \_U1 = \_T1, typename \_U2 = \_T2>  
constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) [pair](#)(\_U1 &&
- template<typename \_U1 = \_T1, typename \_U2 = \_T2>  
constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) [pair](#)(\_U1 &&\_\_x
- template<typename \_U1, typename \_U2>  
constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) [pair](#)([pair](#)<\_U1
- template<typename \_U1, typename \_U2>  
constexpr \_U2 &constexpr **explicit** (!\_S\_convertible<\_U1, \_U2 >()) [pair](#)([pair](#)<\_U1
- constexpr **explicit** (!\_S\_convertible<const \_T1 &, const \_T2 &>()) [pair](#)(const type\_identity\_t<\_T1 > &\_\_x
- template<typename \_U1, typename \_U2>  
constexpr **explicit** (!\_S\_convertible<const \_U1 &, const \_U2 &>()) [pair](#)(const [pair](#)<\_U1
- template<typename \_U1, typename \_U2>  
constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible<const \_U1 &, const \_U2 &>()) [pair](#)(const [pair](#)<\_U1
- template<typename \_U1, typename \_U2>  
constexpr **explicit** (!\_S\_convertible<const \_U1, const \_U2 >()) [pair](#)(const [pair](#)<\_U1
- template<typename \_U1, typename \_U2>  
constexpr \_U2 &constexpr **explicit** (!\_S\_convertible<const \_U1, const \_U2 >()) [pair](#)(const [pair](#)<\_U1
- template<\_\_eligible\_pair\_like<[pair](#)> \_UPair>  
constexpr **explicit** (!\_S\_convertible\_from\_pair\_like<\_UPair >()) [pair](#)(\_UPair &&)=delete
- template<\_\_eligible\_pair\_like<[pair](#)> \_UPair>  
constexpr \_U2 &&constexpr **explicit** (!\_S\_convertible\_from\_pair\_like<\_UPair >()) [pair](#)(\_UPair &&\_\_p)
- constexpr **explicit** (\_\_not<\_\_and<\_\_is\_implicitly\_default\_constructible<\_T1 >, \_\_is\_implicitly\_default\_↵  
constructible<\_T2 > >>()) [pair](#)() noexcept(is\_nothrow\_default\_constructible\_v<\_T1 > &&is\_nothrow\_↵  
default\_constructible\_v<\_T2 >)
- constexpr \_U2 &&constexpr \_U2 &\_\_p **noexcept** (\_S\_nothrow\_constructible<\_U1 &, \_U2 &>())
- constexpr \_U2 &constexpr \_U2 &&\_\_p **noexcept** (\_S\_nothrow\_constructible<\_U1, \_U2 >())
- constexpr \_U2 &&\_\_y **noexcept** (\_S\_nothrow\_constructible<\_U1, \_U2 >())
- constexpr const \_T2 &\_\_y **noexcept** (\_S\_nothrow\_constructible<const \_T1 &, const \_T2 &>())
- constexpr \_U2 &&constexpr \_U2 &\_\_p **noexcept** (\_S\_nothrow\_constructible<const \_U1 &, const \_U2 &>())
- constexpr \_U2 &constexpr \_U2 &&\_\_p **noexcept** (\_S\_nothrow\_constructible<const \_U1, const \_U2 >())
- template<\_\_eligible\_pair\_like<[pair](#)> \_UPair>  
constexpr [pair](#) & **operator**= (\_UPair &&\_\_p)
- template<\_\_eligible\_pair\_like<[pair](#)> \_UPair>  
constexpr const [pair](#) & **operator**= (\_UPair &&\_\_p) const
- [pair](#) & **operator**= (const [pair](#) &)=delete
- constexpr const [pair](#) & **operator**= (const [pair](#) &\_\_p) const
- constexpr [pair](#) & **operator**= (const [pair](#) &\_\_p) noexcept(\_S\_nothrow\_assignable<const \_T1 &, const \_T2 &>())
- template<typename \_U1, typename \_U2>  
requires (\_S\_const\_assignable<const \_U1&, const \_U2&>())  
constexpr const [pair](#) & **operator**= (const [pair](#)<\_U1, \_U2 > &\_\_p) const

- `template<typename _U1, typename _U2>`  
`requires (_S_assignable<const _U1&, const _U2&>())`  
`constexpr pair & operator= (const pair< _U1, _U2 > &__p) noexcept(_S_nothrow_assignable< const _U1 &, const _U2 & >())`
- `constexpr const pair & operator= (pair &&__p) const`
- `constexpr pair & operator= (pair &&__p) noexcept(_S_nothrow_assignable< _T1, _T2 >())`
- `template<typename _U1, typename _U2>`  
`requires (_S_const_assignable<_U1, _U2>())`  
`constexpr const pair & operator= (pair< _U1, _U2 > &&__p) const`
- `template<typename _U1, typename _U2>`  
`requires (_S_assignable<_U1, _U2>())`  
`constexpr pair & operator= (pair< _U1, _U2 > &&__p) noexcept(_S_nothrow_assignable< _U1, _U2 >())`
- `constexpr void swap (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _T1 >, __is_nothrow_swappable< const _T2 > >)`
- `constexpr void swap (pair &__p) noexcept(__and< __is_nothrow_swappable< _T1 >, __is_nothrow_swappable< _T2 > >::value)`

### Public Attributes

- `_T1` *first*
- `_T2` *second*

### Related Symbols

(Note that these are not member symbols.)

- `template<typename _T1, typename _T2>`  
`pair (_T1, _T2) -> pair< _T1, _T2 >`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2, typename _U1, typename _U2>`  
`constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)`
- `template<typename _T1, typename _T2>`  
`constexpr enable_if< __and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`

### 6.777.1 Detailed Description

`template<typename _T1, typename _T2>`

`struct std::pair< _T1, _T2 >`

Struct holding two objects (or references) of arbitrary type.

### Template Parameters

|                  |                                   |
|------------------|-----------------------------------|
| <code>_T1</code> | Type of the <i>first</i> member.  |
| <code>_T2</code> | Type of the <i>second</i> member. |

<https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

### 6.777.2 Member Typedef Documentation

#### first\_type

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the first member.

#### second\_type

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the second member.

### 6.777.3 Constructor & Destructor Documentation

#### pair() [1/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
 const pair< _T1, _T2 > &) [constexpr], [default]
```

Copy constructor.

Referenced by [pair\(\)](#).

#### pair() [2/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
 pair< _T1, _T2 > &&) [constexpr], [default]
```

Move constructor.

### 6.777.4 Member Function Documentation

#### explicit() [1/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1 &, _U2 & > ()) [constexpr]
```

Converting constructor from a non-const pair<U1, U2> lvalue.

#### explicit() [2/7]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2>
std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) && [constexpr]
```

Constructor accepting two values of arbitrary types.

#### explicit() [3/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< _U1, _U2 > ()) [constexpr]
```

Converting constructor from a non-const pair<U1, U2> rvalue.

**explicit()** [4/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _T1 &, const _T2 & > ()) const & [constexpr]
```

Constructor accepting lvalues of first\_type and second\_type.

**explicit()** [5/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _U1 &, const _U2 & > ()) const [constexpr]
```

Converting constructor from a const pair<U1, U2> lvalue.

**explicit()** [6/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
 !_S_convertible< const _U1, const _U2 > ()) const [constexpr]
```

Converting constructor from a const pair<U1, U2> rvalue.

**explicit()** [7/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
 __not_< __and_< __is_implicitly_default_constructible< _T1 >, __is_implicitly_↵
default_constructible< _T2 > > > ()) [inline], [constexpr], [noexcept]
```

Default constructor.

**operator=()** [1/8]

```
template<typename _T1, typename _T2>
const pair & std::pair< _T1, _T2 >::operator= (
 const pair< _T1, _T2 > & __p) const [inline], [constexpr]
```

Copy assignment operator (const).

**operator=()** [2/8]

```
template<typename _T1, typename _T2>
pair & std::pair< _T1, _T2 >::operator= (
 const pair< _T1, _T2 > & __p) [inline], [constexpr], [noexcept]
```

Copy assignment operator.

**operator=()** [3/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<const _U1&, const _U2&>())
const pair & std::pair< _T1, _T2 >::operator= (
 const pair< _U1, _U2 > & __p) const [inline], [constexpr]
```

Converting assignment from a const pair<U1, U2> lvalue.



**operator=()** [4/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<const _U1&, const _U2&>())
pair & std::pair<_T1, _T2>::operator= (
 const pair<_U1, _U2> & __p) [inline], [constexpr], [noexcept]
```

Converting assignment from a const pair<U1, U2> lvalue.

**operator=()** [5/8]

```
template<typename _T1, typename _T2>
const pair & std::pair<_T1, _T2>::operator= (
 pair<_T1, _T2> && __p) const [inline], [constexpr]
```

Move assignment operator (const).

**operator=()** [6/8]

```
template<typename _T1, typename _T2>
pair & std::pair<_T1, _T2>::operator= (
 pair<_T1, _T2> && __p) [inline], [constexpr], [noexcept]
```

Move assignment operator.

**operator=()** [7/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<_U1, _U2>())
const pair & std::pair<_T1, _T2>::operator= (
 pair<_U1, _U2> && __p) const [inline], [constexpr]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

**operator=()** [8/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<_U1, _U2>())
pair & std::pair<_T1, _T2>::operator= (
 pair<_U1, _U2> && __p) [inline], [constexpr], [noexcept]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

**swap()**

```
template<typename _T1, typename _T2>
void std::pair<_T1, _T2>::swap (
 pair<_T1, _T2> & __p) [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

Referenced by `std::pair<const __iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>>::swap()`, `std::sub_match<const char*>` and `std::swap()`.

**6.777.5 Member Data Documentation****first**

```
template<typename _T1, typename _T2>
_T1 std::pair<_T1, _T2>::first
```

The first member.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_find\\_template\(\)](#), [std::\\_\\_sample\(\)](#), [std::set< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator<=>\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator==\(\)](#), and [std::shuffle\(\)](#).

## second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

Referenced by [std::\\_\\_sample\(\)](#), [std::set< \\_Key, \\_Cmp, polymorphic\\_allocator< \\_Key > >::insert\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator<=>\(\)](#), [std::pair< const \\_\\_iter\\_key\\_t< \\_InputIterator >, \\_\\_iter\\_val\\_t< \\_InputIterator > >::operator==\(\)](#), and [std::shuffle\(\)](#).

The documentation for this struct was generated from the following files:

- [bits/stl\\_iterator.h](#)
- [stl\\_pair.h](#)
- [tuple](#)

## 6.778 \_\_gnu\_pbds::detail::pairing\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <pairing_heap.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef [left\\_child\\_next\\_sibling\\_heap\\_node](#)< `Value_Type`, `null_type`, `_Alloc` > **node**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **pairing\_heap** (const Cmp\_Fn &)
- **pairing\_heap** (const pairing\_heap &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- void **erase** (point\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** (pairing\_heap &)
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, pairing\_heap &)
- void **swap** (left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, [null\\_type](#), \_Alloc > &)
- void **swap** (pairing\_heap &)
- const\_reference **top** () const

## Protected Types

- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef alloc\_traits::const\_pointer **node\_const\_pointer**
- typedef [null\\_type](#) **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

## Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 6.778.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

The documentation for this class was generated from the following file:

- [pairing\\_heap.hpp](#)

#### 6.779 \_\_gnu\_pbds::pairing\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::pairing\_heap\_tag:



#### 6.779.1 Detailed Description

Pairing-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 6.780 \_\_gnu\_parallel::parallel\_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:



### Public Member Functions

- [parallel\\_tag](#) ()
- [parallel\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()

- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

### 6.780.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

### 6.780.2 Constructor & Destructor Documentation

#### **parallel\_tag()** [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag () [inline]
```

Default constructor. Use default number of threads.

#### **parallel\_tag()** [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
 _ThreadIndex __num_threads) [inline]
```

Default constructor. Recommend number of threads to use.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

### 6.780.3 Member Function Documentation

#### **\_\_get\_num\_threads()**

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#)

#### **set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline]
```

Set the desired number of threads.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.781 `std::bernoulli_distribution::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `bernoulli_distribution` `distribution_type`

### Public Member Functions

- `param_type` (double \_\_p)
- double `p` () const

### Friends

- bool `operator==` (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.781.1 Detailed Description

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6.782 `std::binomial_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `binomial_distribution<_IntType>` `distribution_type`

### Public Member Functions

- `param_type` (\_IntType \_\_t, double \_\_p=0.5)
- double `p` () const
- \_IntType `t` () const

### Friends

- class `binomial_distribution<_IntType>`
- bool `operator==` (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.782.1 Detailed Description

```
template<typename _IntType = int>
struct std::binomial_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.783 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `cauchy_distribution<_RealType>` `distribution_type`

### Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- \_RealType **a** () const
- \_RealType **b** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.783.1 Detailed Description

```
template<typename _RealType = double>
struct std::cauchy_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.784 std::chi\_squared\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

#### Public Types

- typedef chi\_squared\_distribution< \_RealType > **distribution\_type**

#### Public Member Functions

- **param\_type** (\_RealType \_\_n)
- \_RealType **n** () const

#### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.784.1 Detailed Description

```
template<typename _RealType = double>
struct std::chi_squared_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.785 std::discrete\_distribution< \_IntType >::param\_type Struct Reference

```
#include <random.h>
```

#### Public Types

- typedef discrete\_distribution< \_IntType > **distribution\_type**



## Public Member Functions

- `template<typename _InputIterator>`  
**param\_type** (`_InputIterator __wbegin, _InputIterator __wend`)
- **param\_type** (`const param_type &`)=default
- **param\_type** (`initializer_list< double > __wil`)
- `template<typename _Func>`  
**param\_type** (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- `param_type & operator=` (`const param_type &`)=default
- `std::vector< double > probabilities` () const

## Friends

- class **discrete\_distribution**< `_IntType` >
- bool **operator==** (`const param_type &__p1, const param_type &__p2`)

### 6.785.1 Detailed Description

`template<typename _IntType = int>`  
**struct std::discrete\_distribution**< `_IntType` >::**param\_type**

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.786 std::exponential\_distribution<\_RealType>::param\_type Struct Reference

```
#include <random.h>
```

## Public Types

- typedef [exponential\\_distribution](#)< `_RealType` > **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __lambda`)
- `_RealType lambda` () const

## Friends

- bool **operator==** (`const param_type &__p1, const param_type &__p2`)

### 6.786.1 Detailed Description

`template<typename _RealType = double>`  
**struct std::exponential\_distribution**< `_RealType` >::**param\_type**

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6.787 std::extreme\_value\_distribution<\_RealType>::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef extreme\_value\_distribution< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1.0))
- \_RealType **a** () const
- \_RealType **b** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.787.1 Detailed Description

```
template<typename _RealType = double>
struct std::extreme_value_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.788 std::fisher\_f\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef fisher\_f\_distribution< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_m, \_RealType \_\_n=\_RealType(1))
- \_RealType **m** () const
- \_RealType **n** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.788.1 Detailed Description

```
template<typename _RealType = double>
struct std::fisher_f_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.789 std::gamma\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `gamma_distribution<_RealType>` **distribution\_type**

### Public Member Functions

- **param\_type** (`_RealType __alpha_val, _RealType __beta_val=_RealType(1)`)
- `_RealType alpha` () const
- `_RealType beta` () const

### Friends

- class `gamma_distribution<_RealType>`
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 6.789.1 Detailed Description

```
template<typename _RealType = double>
struct std::gamma_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.790 `std::geometric_distribution<_IntType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `geometric_distribution<_IntType>` **distribution\_type**

### Public Member Functions

- **param\_type** (double \_\_p)
- double **p** () const

### Friends

- class `geometric_distribution<_IntType>`
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

#### 6.790.1 Detailed Description

```
template<typename _IntType = int>
struct std::geometric_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6.791 `std::lognormal_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef lognormal\_distribution< \_RealType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_m, \_RealType \_\_s=\_RealType(1))
- \_RealType **m** () const
- \_RealType **s** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.791.1 Detailed Description

```
template<typename _RealType = double>
struct std::lognormal_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.792 std::negative\_binomial\_distribution< \_IntType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef negative\_binomial\_distribution< \_IntType > **distribution\_type**

### Public Member Functions

- **param\_type** (\_IntType \_\_k, double \_\_p=0.5)
- \_IntType **k** () const
- double **p** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.792.1 Detailed Description

```
template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.793 std::normal\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

## Public Types

- typedef normal\_distribution< \_RealType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_mean, \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

## Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

### 6.793.1 Detailed Description

```
template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6.794 std::piecewise\_constant\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

## Public Types

- typedef piecewise\_constant\_distribution< \_RealType > **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW>  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- **param\_type** (const param\_type &)=default
- template<typename \_Func>  
**param\_type** ([initializer\\_list](#)< \_RealType > \_\_bi, \_Func \_\_fw)
- template<typename \_Func>  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< \_RealType > **intervals** () const
- param\_type & **operator=** (const param\_type &)=default

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &, const [std::piecewise\\_constant\\_distribution](#)< \_RealType1 > &)
- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)
- class **piecewise\_constant\_distribution**< \_RealType >

### 6.794.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_constant_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.795 std::piecewise\_linear\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef piecewise\_linear\_distribution< \_RealType > **distribution\_type**

### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW>  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- **param\_type** (const param\_type &)=default
- template<typename \_Func>  
**param\_type** (initializer\_list< \_RealType > \_\_bl, \_Func \_\_fw)
- template<typename \_Func>  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **std::vector**< double > **densities** () const
- **std::vector**< \_RealType > **intervals** () const
- param\_type & **operator=** (const param\_type &)=default

### Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
**std::basic\_ostream**< \_CharT, \_Traits > & **operator<<** (**std::basic\_ostream**< \_CharT, \_Traits > &, const **std::piecewise\_linear\_distribution**< \_RealType1 > &)
- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)
- class **piecewise\_linear\_distribution**< \_RealType >

### 6.795.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_linear_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.796 std::poisson\_distribution< \_IntType >::param\_type Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `poisson_distribution<_IntType>` **distribution\_type**

### Public Member Functions

- **param\_type** (double \_\_mean)
- double **mean** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)
- class **poisson\_distribution<\_IntType>**

#### 6.796.1 Detailed Description

```
template<typename _IntType = int>
struct std::poisson_distribution<_IntType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 6.797 `std::student_t_distribution<_RealType>::param_type` Struct Reference

```
#include <random.h>
```

### Public Types

- typedef `student_t_distribution<_RealType>` **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_n)
- \_RealType **n** () const

### Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

#### 6.797.1 Detailed Description

```
template<typename _RealType = double>
struct std::student_t_distribution<_RealType>::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.798 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference

```
#include <uniform_int_dist.h>
```

## Public Types

- typedef [uniform\\_int\\_distribution](#)< \_IntType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_IntType \_\_a, \_IntType \_\_b= [\\_\\_gnu\\_cxx::\\_\\_int\\_traits](#)< \_IntType >::\_\_max)
- [result\\_type](#) **a** () const
- [result\\_type](#) **b** () const

## Friends

- bool **operator!=** (const param\_type &\_\_p1, const param\_type &\_\_p2)
- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

### 6.798.1 Detailed Description

```
template<typename _IntType = int>
struct std::uniform_int_distribution< _IntType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [uniform\\_int\\_dist.h](#)

### 6.799 std::uniform\_real\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```

## Public Types

- typedef [uniform\\_real\\_distribution](#)< \_RealType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- [result\\_type](#) **a** () const
- [result\\_type](#) **b** () const

## Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

### 6.799.1 Detailed Description

```
template<typename _RealType = double>
struct std::uniform_real_distribution< _RealType >::param_type
```

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 6.800 std::weibull\_distribution< \_RealType >::param\_type Struct Reference

```
#include <random.h>
```



## Public Types

- typedef weibull\_distribution< \_RealType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1.0))
- \_RealType **a** () const
- \_RealType **b** () const

## Friends

- bool **operator==** (const param\_type &\_\_p1, const param\_type &\_\_p2)

### 6.800.1 Detailed Description

template<typename \_RealType = double>  
 struct std::weibull\_distribution< \_RealType >::param\_type

Parameter type.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 6.801 \_\_gnu\_pbds::detail::pat\_trie\_base Struct Reference

#include <pat\_trie\_base.hpp>

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base:



## Classes

- class [\\_CIter](#)
- struct [\\_Head](#)
- struct [\\_Inode](#)
- class [\\_Iter](#)
- struct [\\_Leaf](#)

- struct [\\_Metadata](#)
- struct [\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_Node\\_base](#)
- class [\\_Node\\_citer](#)
- class [\\_Node\\_iter](#)

## Public Types

- enum [node\\_type](#) { [i\\_node](#) , [leaf\\_node](#) , [head\\_node](#) }

### 6.801.1 Detailed Description

Base type for PATRICIA trees.

### 6.801.2 Member Enumeration Documentation

#### node\_type

enum [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::node\\_type](#)

Three types of nodes.

[i\\_node](#) is used by [\\_Node](#), [leaf\\_node](#) by [\\_Leaf](#), and [head\\_node](#) by [\\_Head](#).

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 6.802 \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > Class Template Reference

```
#include <pat_trie_.hpp>
```

Inheritance diagram for [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#):



## Public Types

- typedef `traits_type::access_traits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `traits_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**

- typedef point\_iterator **iterator**
- typedef traits\_base::key\_const\_pointer **key\_const\_pointer**
- typedef traits\_base::key\_const\_reference **key\_const\_reference**
- typedef traits\_base::key\_pointer **key\_pointer**
- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef traits\_type::node\_const\_iterator **node\_const\_iterator**
- typedef traits\_type::node\_iterator **node\_iterator**
- enum [node\\_type](#) { **i\_node** , **leaf\_node** , **head\_node** }
- typedef traits\_type::node\_update **node\_update**
- typedef traits\_type::const\_iterator **point\_const\_iterator**
- typedef traits\_type::iterator **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef traits\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef [stored\\_data](#)< value\_type, size\_type, Store\_Hash > **stored\_data\_type**
- typedef traits\_base::value\_type **value\_type**

### Public Member Functions

- **pat\_trie\_map** (const access\_traits &)
- **pat\_trie\_map** (const pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- const\_iterator **erase** (const\_iterator)
- const\_reverse\_iterator **erase** (const\_reverse\_iterator)
- iterator **erase** (iterator)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- access\_traits & **get\_access\_traits** ()
- const access\_traits & **get\_access\_traits** () const
- node\_update & **get\_node\_update** ()
- const node\_update & **get\_node\_update** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- void **join** (pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)

- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_end** ()
- node\_const\_iterator **node\_end** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Member Functions

- template<typename It>  
void **copy\_from\_range** (It, It)
- node\_pointer **recursive\_copy\_node** (node\_const\_pointer)
- void **value\_swap** (pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)

### 6.802.1 Detailed Description

template<typename Key, typename Mapped, typename Node\_And\_It\_Traits, typename \_Alloc>  
class **\_\_gnu\_pbds::detail::pat\_trie\_map**< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptsset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000

### 6.802.2 Member Enumeration Documentation

#### node\_type

enum **\_\_gnu\_pbds::detail::pat\_trie\_base::node\_type** [inherited]

Three types of nodes.

i\_node is used by \_Inode, leaf\_node by \_Leaf, and head\_node by \_Head.

### 6.802.3 Member Function Documentation

#### node\_begin() [1/2]

template<typename Key, typename Mapped, typename Node\_And\_It\_Traits, typename \_Alloc>  
pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::node\_iterator **\_\_gnu\_pbds::detail::pat\_trie\_map**<  
Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::node\_begin () [inline]

Returns a node\_iterator corresponding to the node at the root of the tree.

**node\_begin()** [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

**node\_end()** [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

**node\_end()** [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_.hpp](#)

**6.803 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::pat_trie_tag`:



### 6.803.1 Detailed Description

PATRICIA trie.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.804 `std::experimental::filesystem::v1::path` Class Reference

```
#include <fs_path.h>
```

### Classes

- class [iterator](#)

### Public Types

- typedef [iterator](#) **const\_iterator**
- typedef [std::basic\\_string](#)< value\_type > **string\_type**
- typedef char **value\_type**

## Public Member Functions

- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path<\_InputIterator, \_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path<\_InputIterator, \_InputIterator>, typename \_Require2 = \_\_detail::\_value\_type\_is\_char<\_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [locale](#) &\_\_loc)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>>>  
**path** (\_Source const &\_\_source)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>, typename \_Require2 = \_\_detail::\_value\_type\_is\_char<\_Source>>>  
**path** (\_Source const &\_\_source, const [locale](#) &\_\_loc)
- **path** (const path &\_\_p)
- **path** (path &&\_\_p) noexcept
- **path** (string\_type &&\_\_source)
- template<typename \_InputIterator>  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **append** (\_Source const &\_\_source)
- template<typename \_InputIterator>  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **assign** (\_Source const &\_\_source)
- path & **assign** (string\_type &&\_\_source)
- [iterator](#) **begin** () const noexcept
- const value\_type \* **c\_str** () const noexcept
- void **clear** () noexcept
- int **compare** (const [basic\\_string\\_view](#)<value\_type> \_\_s) const
- int **compare** (const path &\_\_p) const noexcept
- int **compare** (const string\_type &\_\_s) const
- int **compare** (const value\_type \* \_\_s) const
- template<typename \_InputIterator>  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **concat** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **concat** (\_Source const &\_\_x)
- bool **empty** () const noexcept
- [iterator](#) **end** () const noexcept
- path **extension** () const
- path **filename** () const
- [std::string](#) **generic\_string** () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>>  
[std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **generic\_string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::u16string](#) **generic\_u16string** () const
- [std::u32string](#) **generic\_u32string** () const
- [std::string](#) **generic\_u8string** () const
- [std::wstring](#) **generic\_wstring** () const
- bool **has\_extension** () const
- bool **has\_filename** () const
- bool **has\_parent\_path** () const
- bool **has\_relative\_path** () const
- bool **has\_root\_directory** () const
- bool **has\_root\_name** () const

- `bool has_root_path () const`
- `bool has_stem () const`
- `bool is_absolute () const`
- `bool is_relative () const`
- `path & make_preferred ()`
- `const string_type & native () const noexcept`
- `operator string_type () const`
- `template<typename _CharT>`  
`__detail::_Path< _CharT *, _CharT * > & operator+= (_CharT __x)`
- `template<typename _Source>`  
`__detail::_Path< _Source > & operator+= (_Source const &__x)`
- `path & operator+= (basic_string_view< value_type > __x)`
- `path & operator+= (const path &__x)`
- `path & operator+= (const string_type &__x)`
- `path & operator+= (const value_type *__x)`
- `path & operator+= (value_type __x)`
- `template<typename _Source>`  
`__detail::_Path< _Source > & operator/= (_Source const &__source)`
- `path & operator/= (const path &__p)`
- `template<typename _Source>`  
`__detail::_Path< _Source > & operator= (_Source const &__source)`
- `path & operator= (const path &__p)`
- `path & operator= (path &&__p) noexcept`
- `path & operator= (string_type &&__source)`
- `path parent_path () const`
- `path relative_path () const`
- `path & remove_filename ()`
- `path & replace_extension (const path &__replacement=path())`
- `path & replace_filename (const path &__replacement)`
- `path root_directory () const`
- `path root_name () const`
- `path root_path () const`
- `path stem () const`
- `std::string string () const`
- `template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>`  
`std::basic_string< _CharT, _Traits, _Allocator > string (const _Allocator &__a=_Allocator()) const`
- `void swap (path &__rhs) noexcept`
- `std::u16string u16string () const`
- `std::u32string u32string () const`
- `std::string u8string () const`
- `std::wstring wstring () const`

## Static Public Attributes

- `static constexpr value_type preferred_separator`

### 6.804.1 Detailed Description

A filesystem path.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)



## 6.805 std::filesystem::path Class Reference

```
#include <filesystem>
```

### Classes

- class [iterator](#)

### Public Types

- using **const\_iterator**
- enum **format**: unsigned char { **native\_format** , **generic\_format** , **auto\_format** }
- using **string\_type**
- using **value\_type**

### Public Member Functions

- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>, typename \_Req2 = \_\_detail::\_value\_type\_<↔ is\_char<\_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [locale](#) &\_\_loc, **format**=auto\_format)
- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, **format**=auto\_format)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>>>  
**path** (\_Source const &\_\_source, **format**=auto\_format)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>, typename \_Require2 = \_\_detail::\_value\_type\_is\_char<↔ \_Source>>>  
**path** (\_Source const &\_\_src, const [locale](#) &\_\_loc, **format**=auto\_format)
- **path** (const path &\_\_p)=default
- **path** (path &&\_\_p) noexcept
- **path** (string\_type &&\_\_source, **format**=auto\_format)
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **append** (\_Source const &\_\_source)
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **assign** (\_Source const &\_\_source)
- path & **assign** (string\_type &&\_\_source)
- [iterator](#) **begin** () const noexcept
- const value\_type \* **c\_str** () const noexcept
- void **clear** () noexcept
- int **compare** ([basic\\_string\\_view](#)< value\_type > \_\_s) const noexcept
- int **compare** (const path &\_\_p) const noexcept
- int **compare** (const string\_type &\_\_s) const noexcept
- int **compare** (const value\_type \* \_\_s) const noexcept
- template<typename \_InputIterator>  
\_\_detail::\_Path2<\_InputIterator> & **concat** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **concat** (\_Source const &\_\_x)
- bool **empty** () const noexcept
- [iterator](#) **end** () const noexcept
- path **extension** () const

- path **filename** () const
- [std::string](#) **generic\_string** () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>  
[std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **generic\_string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::u16string](#) **generic\_u16string** () const
- [std::u32string](#) **generic\_u32string** () const
- [std::string](#) **generic\_u8string** () const
- [std::wstring](#) **generic\_wstring** () const
- bool **has\_extension** () const noexcept
- bool **has\_filename** () const noexcept
- bool **has\_parent\_path** () const noexcept
- bool **has\_relative\_path** () const noexcept
- bool **has\_root\_directory** () const noexcept
- bool **has\_root\_name** () const noexcept
- bool **has\_root\_path** () const noexcept
- bool **has\_stem** () const noexcept
- bool **is\_absolute** () const noexcept
- bool **is\_relative** () const noexcept
- path **lexically\_normal** () const
- path **lexically\_proximate** (const path &base) const
- path **lexically\_relative** (const path &base) const
- path & **make\_preferred** ()
- const string\_type & **native** () const noexcept
- **operator string\_type** () const
- template<typename \_CharT>  
\_\_detail::\_Path2<\_CharT \* > & **operator+=** (\_CharT \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator+=** (\_Source const &\_\_x)
- path & **operator+=** ([basic\\_string\\_view](#)<value\_type> \_\_x)
- path & **operator+=** (const path &\_\_x)
- path & **operator+=** (const string\_type &\_\_x)
- path & **operator+=** (const value\_type \*\_\_x)
- path & **operator+=** (value\_type \_\_x)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator/=** (\_Source const &\_\_source)
- path & **operator/=** (const path &\_\_p)
- template<typename \_Source>  
\_\_detail::\_Path<\_Source> & **operator=** (\_Source const &\_\_source)
- path & **operator=** (const path &)
- path & **operator=** (path &&) noexcept
- path & **operator=** (string\_type &&\_\_source)
- path **parent\_path** () const
- path **relative\_path** () const
- path & **remove\_filename** ()
- path & **replace\_extension** (const path &\_\_replacement=path())
- path & **replace\_filename** (const path &\_\_replacement)
- path **root\_directory** () const
- path **root\_name** () const
- path **root\_path** () const
- path **stem** () const
- [std::string](#) **string** () const

- template<typename \_CharT, typename \_Traits, typename \_Allocator>  
basic\_string< \_CharT, \_Traits, \_Allocator > **string** (const \_Allocator &\_\_a) const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>  
std::basic\_string< \_CharT, \_Traits, \_Allocator > **string** (const \_Allocator &\_\_a=\_Allocator()) const
- void **swap** (path &\_\_rhs) noexcept
- std::u16string **u16string** () const
- std::u32string **u32string** () const
- std::string **u8string** () const
- std::wstring **wstring** () const

### Static Public Attributes

- static constexpr value\_type **preferred\_separator**

### Friends

- path **operator/** (const path &\_\_lhs, const path &\_\_rhs)
- template<typename \_CharT, typename \_Traits>  
std::basic\_ostream< \_CharT, \_Traits > & **operator<<** (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const path &\_\_p)
- strong\_ordering **operator<=>** (const path &\_\_lhs, const path &\_\_rhs) noexcept
- bool **operator==** (const path &\_\_lhs, const path &\_\_rhs) noexcept
- template<typename \_CharT, typename \_Traits>  
std::basic\_istream< \_CharT, \_Traits > & **operator>>** (std::basic\_istream< \_CharT, \_Traits > &\_\_is, path &\_\_p)

### Related Symbols

(Note that these are not member symbols.)

- void **swap** (path &\_\_lhs, path &\_\_rhs) noexcept
- template<typename \_InputIterator, typename \_Require = \_\_detail::\_Path2<\_InputIterator>, typename \_CharT = \_\_detail::\_value\_type\_is\_char\_or\_char8\_t<\_InputIterator>>  
path **u8path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source, typename \_Require = \_\_detail::\_Path<\_Source>, typename \_CharT = \_\_detail::\_value\_type\_is\_char\_or\_char8\_t<\_Source>>  
path **u8path** (const \_Source &\_\_source)

#### 6.805.1 Detailed Description

A filesystem path.

Since

C++17

#### 6.805.2 Member Enumeration Documentation

##### format

enum **std::filesystem::path::format** : unsigned char  
path::format is ignored in this implementation

### 6.805.3 Friends And Related Symbol Documentation

#### **operator/**

```
path operator/ (
 const path & __lhs,
 const path & __rhs) [friend]
```

Append one path to another.

#### **operator<<**

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const path & __p) [friend]
```

Write a path to a stream.

#### **operator<=>**

```
strong_ordering operator<=> (
 const path & __lhs,
 const path & __rhs) [friend]
```

Compare paths.

#### **operator==**

```
bool operator== (
 const path & __lhs,
 const path & __rhs) [friend]
```

Compare paths.

#### **operator>>**

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 path & __p) [friend]
```

Read a path from a stream.

References [std::move\(\)](#).

The documentation for this class was generated from the following file:

- [bits/fs\\_path.h](#)

## 6.806 std::philox\_engine< \_UIntType, \_\_w, \_\_n, \_\_r, \_\_consts > Class Template Reference

```
#include <random>
```

### Public Types

- using **result\_type**

### Public Member Functions

- template<typename \_Sseq>  
[philox\\_engine](#) (\_Sseq &\_\_q)
- **philox\_engine** (result\_type \_\_value)

- void **discard** (unsigned long long \_\_z)
- result\_type **operator()** ()
- template<typename \_Sseq>  
requires \_\_is\_seed\_seq<\_Sseq>  
void **seed** (\_Sseq &\_\_q)
- void **seed** (result\_type \_\_value=default\_seed)
- void **set\_counter** (const array< result\_type, \_\_n > &\_\_counter)

### Static Public Member Functions

- static constexpr result\_type **max** ()
- static constexpr result\_type **min** ()

### Static Public Attributes

- static constexpr result\_type **default\_seed**
- static constexpr array< result\_type, \_\_n/2 > **multipliers**
- static constexpr array< result\_type, \_\_n/2 > **round\_consts**
- static constexpr size\_t **round\_count**
- static constexpr size\_t **word\_count**
- static constexpr size\_t **word\_size**

### Friends

- template<typename \_CharT, typename \_Traits>  
**basic\_ostream**< \_CharT, \_Traits > & **operator<<** (**basic\_ostream**< \_CharT, \_Traits > &\_\_os, const philox\_engine &\_\_x)
- bool **operator==** (const philox\_engine &, const philox\_engine &)=default
- template<typename \_CharT, typename \_Traits>  
**basic\_istream**< \_CharT, \_Traits > & **operator>>** (**basic\_istream**< \_CharT, \_Traits > &\_\_is, philox\_engine &\_\_x)

#### 6.806.1 Detailed Description

**template<typename \_UIntType, size\_t \_\_w, size\_t \_\_n, size\_t \_\_r, \_UIntType... \_\_consts>**  
**class std::philox\_engine< \_UIntType, \_\_w, \_\_n, \_\_r, \_\_consts >**

A discrete pseudorandom number generator with weak cryptographic properties.

This algorithm was designed to be used for highly parallel random number generation, and is capable of immensely long periods. It provides "Crush-resistance", denoting an ability to pass the TestU01 Suite's "Big Crush" test, demonstrating significant apparent entropy.

It is not intended for cryptographic use and should not be used for such, despite being based on cryptographic primitives. The typedefs `philox4x32` and `philox4x64` are provided as suitable defaults for most use cases, providing high-quality random numbers with reasonable performance.

This algorithm was created by John Salmon, Mark Moraes, Ron Dror, and David Shaw as a product of D.E. Shaw Research.

### Template Parameters

|                  |             |
|------------------|-------------|
| <code>__w</code> | Word size   |
| <code>__n</code> | Buffer size |

|                       |                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>__r</code>      | Rounds                                                                                                                               |
| <code>__consts</code> | Multiplication and round constant pack, ordered as $M_{\{0\}}, C_{\{0\}}, M_{\{1\}}, C_{\{1\}}, \dots, M_{\{N/2-1\}}, C_{\{N/2-1\}}$ |

Since

C++26

## 6.806.2 Constructor & Destructor Documentation

### `philox_engine()`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
template<typename _Sseq>
std::philox_engine< _UIntType, __w, __n, __r, __consts >::philox_engine (
 _Sseq & __q) [inline], [explicit]
seed sequence constructor for philox_engine
```

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__q</code> | the seed sequence |
|------------------|-------------------|

## 6.806.3 Member Function Documentation

### `discard()`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::discard (
 unsigned long long __z) [inline]
discards __z numbers
```

#### Parameters

|                  |                                 |
|------------------|---------------------------------|
| <code>__z</code> | number of iterations to discard |
|------------------|---------------------------------|

### `max()`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
constexpr result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::max () [inline],
[static], [constexpr]
```

The maximum value that this engine can return.

Referenced by `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>`,  
and `std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::set__counter()`.

**min()**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
constexpr result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::min () [inline],
[static], [constexpr]
```

The minimum value that this engine can return.

**operator>()()**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::operator() () [inline]
```

outputs a single w-bit number and handles state advancement

**Returns**

return\_type

**seed()**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
requires __is_seed_seq<_Sseq>
template<typename _Sseq>
requires __is_seed_seq<_Sseq>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::seed (
 _Sseq & __q)
```

seeds philox\_engine by seed sequence

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__q</code> | the seed sequence |
|------------------|-------------------|

References [std::max\(\)](#).

**set\_counter()**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::set_counter (
 const array< result_type, __n > & __counter) [inline]
```

sets the internal counter "cleartext"

**Parameters**

|                        |                     |
|------------------------|---------------------|
| <code>__counter</code> | std::array of len N |
|------------------------|---------------------|

**6.806.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const philox_engine< _UIntType, __w, __n, __r, __consts > & __x) [friend]
```

outputs the state of the generator

#### Parameters

|                   |                                               |
|-------------------|-----------------------------------------------|
| <code>__os</code> | An output stream.                             |
| <code>__x</code>  | A <code>philox_engine</code> object reference |

#### Returns

the state of the Philox Engine in `__os`

#### `operator==`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
bool operator== (
 const philox_engine< _UIntType, __w, __n, __r, __consts > & ,
 const philox_engine< _UIntType, __w, __n, __r, __consts > &) [friend]
```

compares two `philox_engine` objects

#### Returns

true if the objects will produce an identical stream, false otherwise

#### `operator>>`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & operator>> (
 basic_istream< _CharT, _Traits > & __is,
 philox_engine< _UIntType, __w, __n, __r, __consts > & __x) [friend]
```

takes input to set the state of the `philox_engine` object

#### Parameters

|                   |                                               |
|-------------------|-----------------------------------------------|
| <code>__is</code> | An input stream.                              |
| <code>__x</code>  | A <code>philox_engine</code> object reference |

#### Returns

`philox_engine` object is set with values from instream

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.807 `std::piecewise_constant_distribution<_RealType >` Class Template Reference

```
#include <random>
```



## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW>  
**piecewise\_constant\_distribution** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- **piecewise\_constant\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_Func>  
**piecewise\_constant\_distribution** ([initializer\\_list](#)< \_RealType > \_\_bl, \_Func \_\_fw)
- template<typename \_Func>  
**piecewise\_constant\_distribution** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- **std::vector**< double > **densities** () const
- **std::vector**< \_RealType > **intervals** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::piecewise\\_constant\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [piecewise\\_constant\\_distribution](#) &\_\_d1, const [piecewise\\_constant\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::piecewise\\_constant\\_distribution](#)< \_RealType1 > &\_\_x)

### 6.807.1 Detailed Description

**template**<typename \_RealType = double>  
**class** [std::piecewise\\_constant\\_distribution](#)<\_RealType>

A [piecewise\\_constant\\_distribution](#) random number distribution.

This distribution produces random numbers  $x, b_0 \leq x < b_n$ , uniformly distributed over each subinterval  $[b_i, b_{i+1})$  according to the probability mass function

$$p(x|b_0, \dots, b_n, \rho_0, \dots, \rho_{n-1}) = \rho_i \cdot \frac{b_{i+1} - x}{b_{i+1} - b_i} + \rho_{i+1} \cdot \frac{x - b_i}{b_{i+1} - b_i}$$

for  $b_i \leq x < b_{i+1}$ .

Since

C++11

## 6.807.2 Member Typedef Documentation

### result\_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

## 6.807.3 Member Function Documentation

### densities()

```
template<typename _RealType = double>
std::vector< double > std::piecewise_constant_distribution< _RealType >::densities () const
[inline]
```

Returns a vector of the probability densities.

### intervals()

```
template<typename _RealType = double>
std::vector< _RealType > std::piecewise_constant_distribution< _RealType >::intervals () const
[inline]
```

Returns a vector of the intervals.

### max()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

### min()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

### operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_constant_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param()** [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_constant_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

**param()** [2/2]

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

**6.807.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::piecewise_constant_distribution< _RealType1 > & __x) [friend]
```

Inserts a piecewise\_constant\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__os</code> | An output stream.                                             |
| <code>__x</code>  | A piecewise_constant_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
 const piecewise_constant_distribution< _RealType > & __d1,
 const piecewise_constant_distribution< _RealType > & __d2) [friend]
```

Return true if two piecewise constant distributions have the same parameters.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::piecewise_constant_distribution< _RealType1 > & __x) [friend]
```

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                               |
| <code>__x</code>  | A <code>piecewise_constant_distribution</code> random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.808 std::piecewise\_construct\_t Struct Reference**

```
#include <stl_pair.h>
```

**6.808.1 Detailed Description**

Tag type for piecewise construction of `std::pair` objects.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

**6.809 std::piecewise\_linear\_distribution< \_RealType > Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef `_RealType` [result\\_type](#)

**Public Member Functions**

- template<typename `_InputIteratorB`, typename `_InputIteratorW`>  
**piecewise\_linear\_distribution** (`_InputIteratorB` `__bfirst`, `_InputIteratorB` `__bend`, `_InputIteratorW` `__wbeg`)
- **piecewise\_linear\_distribution** (const [param\\_type](#) & `__p`)
- template<typename `_Func`>  
**piecewise\_linear\_distribution** ([initializer\\_list](#)< `_RealType` > `__bl`, `_Func` `__fw`)

- template<typename \_Func>  
**piecewise\_linear\_distribution** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< \_RealType > **intervals** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::piecewise\\_linear\\_distribution](#)< \_RealType1 > &\_\_x)
- bool **operator==** (const [piecewise\\_linear\\_distribution](#) &\_\_d1, const [piecewise\\_linear\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::piecewise\\_linear\\_distribution](#)< \_RealType1 > &\_\_x)

## 6.809.1 Detailed Description

**template<typename \_RealType = double>**  
**class std::piecewise\_linear\_distribution<\_RealType>**

A `piecewise_linear_distribution` random number distribution.

This distribution produces random numbers  $x$ ,  $b_0 \leq x < b_n$ , distributed over each subinterval  $[b_i, b_{i+1})$  according to the probability mass function  $p(x|b_0, \dots, b_n, \rho_0, \dots, \rho_n) = \rho_i$ , for  $b_i \leq x < b_{i+1}$ .

Since

C++11

## 6.809.2 Member Typedef Documentation

### result\_type

template<typename \_RealType = double>  
typedef \_RealType [std::piecewise\\_linear\\_distribution](#)< \_RealType >::result\_type  
The type of the range of the distribution.

### 6.809.3 Member Function Documentation

#### **densities()**

```
template<typename _RealType = double>
```

```
std::vector< double > std::piecewise_linear_distribution< _RealType >::densities () const [inline]
```

Return a vector of the probability densities of the distribution.

#### **intervals()**

```
template<typename _RealType = double>
```

```
std::vector< _RealType > std::piecewise_linear_distribution< _RealType >::intervals () const [inline]
```

Return the intervals of the distribution.

#### **max()**

```
template<typename _RealType = double>
```

```
result_type std::piecewise_linear_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### **min()**

```
template<typename _RealType = double>
```

```
result_type std::piecewise_linear_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### **operator()()**

```
template<typename _RealType = double>
```

```
template<typename _UniformRandomNumberGenerator>
```

```
result_type std::piecewise_linear_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### **param()** [1/2]

```
template<typename _RealType = double>
```

```
param_type std::piecewise_linear_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### **param()** [2/2]

```
template<typename _RealType = double>
```

```
void std::piecewise_linear_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### **Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::piecewise_linear_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

**6.809.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::piecewise_linear_distribution<_RealType1> & __x) [friend]
```

Inserts a piecewise\_linear\_distribution random number distribution \_\_x into the output stream \_\_os.

**Parameters**

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__os</code> | An output stream.                                           |
| <code>__x</code>  | A piecewise_linear_distribution random number distribution. |

**Returns**

The output stream with the state of \_\_x inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
 const piecewise_linear_distribution<_RealType> & __d1,
 const piecewise_linear_distribution<_RealType> & __d2) [friend]
```

Return true if two piecewise linear distributions have the same parameters.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits> & operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::piecewise_linear_distribution<_RealType1> & __x) [friend]
```

Extracts a piecewise\_linear\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                |
| <code>__x</code>  | A piecewise_linear_distribution random number generator engine. |

**Returns**

The input stream with `___x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.810 std::plus<\_Tp> Struct Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for `std::plus<_Tp>`:

**Public Types**

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

**Public Member Functions**

- constexpr `_Tp` [operator\(\)](#) (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

**6.810.1 Detailed Description**

```
template<typename _Tp>
struct std::plus<_Tp>
```

One of the [math functors](#).

**6.810.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```



**result\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, \_Tp >::result\_type [inherited]  
 result\_type is the return type

**second\_argument\_type**

typedef \_Tp [std::binary\\_function](#)< \_Tp, \_Tp, \_Tp >::second\_argument\_type [inherited]  
 second\_argument\_type is the type of the second argument

**6.810.3 Member Function Documentation****operator>()()**

```
template<typename _Tp>
_Tp std::plus< _Tp >::operator() (
 const _Tp & __x,
 const _Tp & __y) const [inline], [constexpr]
```

Returns the sum.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

**6.811 \_\_gnu\_pbds::point\_invalidation\_guarantee Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::point\_invalidation\_guarantee:

**6.811.1 Detailed Description**

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased,

regardless of modifications to the container object.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.812 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- **`pointer_to_binary_function`** (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result` **`operator()`** (`_Arg1 __x, _Arg2 __y`) const

### Protected Attributes

- `_Result(* _M_ptr)(_Arg1, _Arg2)`

#### 6.812.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

### 6.812.2 Member Typedef Documentation

#### first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

#### result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type [inherited]
result_type is the return type
```

#### second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:
```

- [stl\\_function.h](#)

## 6.813 std::pointer\_to\_unary\_function< \_Arg, \_Result > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::pointer\_to\_unary\_function< \_Arg, \_Result >:



### Public Types

- typedef \_Arg [argument\\_type](#)
- typedef \_Result [result\\_type](#)

### Public Member Functions

- **pointer\_to\_unary\_function** (\_Result(\*\_\_x)(\_Arg))
- \_Result **operator()** (\_Arg \_\_x) const

## Protected Attributes

- `_Result(*_M_ptr)(_Arg)`

### 6.813.1 Detailed Description

```
template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

### 6.813.2 Member Typedef Documentation

#### argument\_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 6.814 std::pointer\_traits<\_Ptr> Struct Template Reference

```
#include <memory>
```

### Public Types

- using [difference\\_type](#)
- using [element\\_type](#)
- using [pointer](#)
- template<typename \_Up>  
using [rebind](#)

### Static Public Member Functions

- static constexpr pointer [pointer\\_to](#) (element\_type &\_\_r)

### 6.814.1 Detailed Description

```
template<typename _Ptr>
struct std::pointer_traits<_Ptr >
```

Uniform interface to all pointer-like types.

Since

C++11

### 6.814.2 Member Typedef Documentation

#### difference\_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::difference_type [inherited]
```

The type used to represent the difference between two pointers.

#### element\_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::element_type [inherited]
```

The type pointed to.

#### pointer

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::pointer [inherited]
```

The pointer type.

#### rebind

```
template<typename _Ptr, typename _Elt>
template<typename _Up>
using std::__ptr_traits_impl< _Ptr, _Elt >::rebind [inherited]
```

A pointer to a different type.

### 6.814.3 Member Function Documentation

#### pointer\_to()

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
 element_type & __r) [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

#### Parameters

|          |                                               |
|----------|-----------------------------------------------|
| ↩        | A reference to an object of type element_type |
| ↩        |                                               |
| ↩        |                                               |
| ↩        |                                               |
| <i>r</i> |                                               |

#### Returns

pointer::pointer\_to(\_\_r)

#### Precondition

pointer::pointer\_to(\_\_r) is a valid expression.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

## 6.815 `std::pointer_traits<_Tp*>` Struct Template Reference

```
#include <memory>
```

### Public Types

- typedef ptrdiff\_t [difference\\_type](#)
- using [difference\\_type](#)
- typedef \_Tp [element\\_type](#)
- using [element\\_type](#)
- typedef \_Tp\* [pointer](#)
- using [pointer](#)
- template<typename \_Up>  
using [rebind](#)
- using [rebind](#)

### Static Public Member Functions

- static constexpr [pointer](#) [pointer\\_to](#) ([element\\_type](#) &\_\_r)
- static constexpr [pointer](#) [pointer\\_to](#) ([element\\_type](#) &\_\_r)
- static constexpr [pointer](#) [pointer\\_to](#) ([element\\_type](#) &\_\_r)

#### 6.815.1 Detailed Description

```
template<typename _Tp>
struct std::pointer_traits<_Tp*>
```

Partial specialization for built-in pointers.

Since

C++11

#### 6.815.2 Member Typedef Documentation

##### **difference\_type** [1/2]

```
template<typename _Tp>
typedef ptrdiff_t std::pointer_traits<_Tp*>::difference_type
Type used to represent the difference between two pointers.
```

##### **difference\_type** [2/2]

```
using std::__ptr_traits_impl<_Tp*, _Elt>::difference_type
The type used to represent the difference between two pointers.
```

##### **element\_type** [1/2]

```
template<typename _Tp>
typedef _Tp std::pointer_traits<_Tp*>::element_type
The type pointed to.
```

##### **element\_type** [2/2]

```
using std::__ptr_traits_impl<_Tp*, _Elt>::element_type
The type pointed to.
```

**pointer** [1/2]

```
template<typename _Tp>
typedef _Tp* std::pointer_traits< _Tp * >::pointer
```

The pointer type.

**pointer** [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::pointer
```

The pointer type.

**rebind** [1/2]

```
template<typename _Tp>
template<typename _Up>
using std::pointer_traits< _Tp * >::rebind
```

A pointer to a different type.

**rebind** [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::rebind
```

A pointer to a different type.

**6.815.3 Member Function Documentation****pointer\_to()** [1/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
 element_type & __r) [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

**Parameters**

|          |                                                            |
|----------|------------------------------------------------------------|
| ↩        | A reference to an object of type <code>element_type</code> |
| ↩        |                                                            |
| ↩        |                                                            |
| ↩        |                                                            |
| <i>r</i> |                                                            |

**Returns**

```
pointer::pointer_to(__r)
```

**Precondition**

```
pointer::pointer_to(__r) is a valid expression.
```

**pointer\_to()** [2/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
 element_type & __r) [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

**Parameters**

|          |                                                            |
|----------|------------------------------------------------------------|
| ↩        | A reference to an object of type <code>element_type</code> |
| ↩        |                                                            |
| ↩        |                                                            |
| ↩        |                                                            |
| <i>r</i> |                                                            |

**Returns**

`pointer::pointer_to(__r)`

**Precondition**

`pointer::pointer_to(__r)` is a valid expression.

**pointer\_to()** [3/3]

```
constexpr pointer std::__ptr_traits_ptr_to< _Tp *, _Elt, bool >::pointer_to (
 element_type & __r) [inline], [static], [constexpr]
```

Obtain a pointer to an object.

**Parameters**

|          |                                                            |
|----------|------------------------------------------------------------|
| ↩        | A reference to an object of type <code>element_type</code> |
| ↩        |                                                            |
| ↩        |                                                            |
| ↩        |                                                            |
| <i>r</i> |                                                            |

**Returns**

`pointer::pointer_to(__r)`

**Precondition**

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

**6.816 std::poisson\_distribution< \_IntType > Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef `_IntType` [result\\_type](#)



## Public Member Functions

- **poisson\_distribution** (const [param\\_type](#) &\_\_p)
- **poisson\_distribution** (double \_\_mean)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **max** () const
- double **mean** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::poisson\\_distribution](#)< \_IntType1 > &\_\_x)
- bool **operator==** (const [poisson\\_distribution](#) &\_\_d1, const [poisson\\_distribution](#) &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::poisson\\_distribution](#)< \_IntType1 > &\_\_x)

### 6.816.1 Detailed Description

**template<typename \_IntType = int>**  
**class std::poisson\_distribution< \_IntType >**

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Since

C++11

### 6.816.2 Member Typedef Documentation

#### **result\_type**

template<typename \_IntType = int>  
 typedef \_IntType [std::poisson\\_distribution](#)< \_IntType >::result\_type

The type of the range of the distribution.

### 6.816.3 Member Function Documentation

#### max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### mean()

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean () const [inline]
```

Returns the distribution parameter mean.

#### min()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::poisson_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

#### operator>() [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

A rejection algorithm when mean  $\geq 12$  and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_FUNCS` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

References [std::abs\(\)](#), [std::numeric\\_limits< \\_Tp >::epsilon\(\)](#), [std::log\(\)](#), and [std::numeric\\_limits< \\_Tp >::max\(\)](#).

#### param() [1/2]

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

#### param() [2/2]

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset () [inline]
Resets the distribution state.
```

**6.816.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::poisson_distribution< _IntType1 > & __x) [friend]
Inserts a poisson_distribution random number distribution __x into the output stream __os.
```

**Parameters**

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <code>__os</code> | An output stream.                                  |
| <code>__x</code>  | A poisson_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _IntType = int>
bool operator== (
 const poisson_distribution< _IntType > & __d1,
 const poisson_distribution< _IntType > & __d2) [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::poisson_distribution< _IntType1 > & __x) [friend]
```

Extracts a poisson\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

|                                                                                                                        |                                                        |
|------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <a href="#"></a><br><code>__is</code> | An input stream.                                       |
| <a href="#"></a><br><code>__x</code>  | A poisson_distribution random number generator engine. |

**Returns**

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.817 std::pmr::polymorphic\_allocator<\_Tp> Class Template Reference**

```
#include <memory_resource>
```

**Public Types**

- using **value\_type**

**Public Member Functions**

- **polymorphic\_allocator** (const polymorphic\_allocator &\_\_other)=default
- template<typename \_Up>  
  **polymorphic\_allocator** (const polymorphic\_allocator< \_Up > &\_\_x) noexcept
- **polymorphic\_allocator** ([memory\\_resource](#) \*\_\_r) noexcept
- **\_Tp \* allocate** (size\_t \_\_n)
- template<typename \_Tp1, typename... \_Args>  
  void **construct** (\_Tp1 \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (\_Tp \*\_\_p, size\_t \_\_n) noexcept
- template<typename \_Up>  
  void **destroy** (\_Up \*\_\_p)
- polymorphic\_allocator & **operator=** (const polymorphic\_allocator &)=delete
- [memory\\_resource](#) \* **resource** () const noexcept
- polymorphic\_allocator **select\_on\_container\_copy\_construction** () const noexcept

**Friends**

- bool **operator==** (const polymorphic\_allocator &\_\_a, const polymorphic\_allocator &\_\_b) noexcept

**6.817.1 Detailed Description**

```
template<typename _Tp>
class std::pmr::polymorphic_allocator< _Tp >
```

Class template polymorphic\_allocator.

Since

C++17

The documentation for this class was generated from the following file:

- [memory\\_resource.h](#)

## 6.818 std::pmr::pool\_options Struct Reference

```
#include <memory_resource>
```

### Public Attributes

- `size_t` **largest\_required\_pool\_block**
- `size_t` [max\\_blocks\\_per\\_chunk](#)

### 6.818.1 Detailed Description

Parameters for tuning a pool resource's behaviour.

Since

C++17

### 6.818.2 Member Data Documentation

#### **max\_blocks\_per\_chunk**

`size_t` `std::pmr::pool_options::max_blocks_per_chunk`

Upper limit on number of blocks in a chunk.

A lower value prevents allocating huge chunks that could remain mostly unused, but means pools will need to replenished more frequently.

The documentation for this struct was generated from the following file:

- [memory\\_resource](#)

## 6.819 \_\_gnu\_pbds::priority\_queue<\_Tv, Cmp\_Fn, Tag, \_Alloc > Class Template Reference

```
#include <priority_queue.hpp>
```

### Public Types

- `typedef _Alloc` **allocator\_type**
- `typedef Cmp_Fn` **cmp\_fn**
- `typedef base_type::const_iterator` **const\_iterator**
- `typedef __rebind_va::const_pointer` **const\_pointer**
- `typedef __rebind_va::const_reference` **const\_reference**
- `typedef Tag` **container\_category**
- `typedef allocator_type::difference_type` **difference\_type**
- `typedef base_type::iterator` **iterator**
- `typedef base_type::point_const_iterator` **point\_const\_iterator**
- `typedef base_type::point_iterator` **point\_iterator**
- `typedef __rebind_va::pointer` **pointer**
- `typedef __rebind_va::reference` **reference**
- `typedef allocator_type::size_type` **size\_type**
- `typedef _Tv` **value\_type**

## Public Member Functions

- [priority\\_queue](#) (const cmp\_fn &r\_cmp\_fn)
- [priority\\_queue](#) (const priority\_queue &other)
- `template<typename It>`  
[priority\\_queue](#) (It first\_it, It last\_it)
- `template<typename It>`  
[priority\\_queue](#) (It first\_it, It last\_it, const cmp\_fn &r\_cmp\_fn)
- `priority_queue & operator=` (const priority\_queue &other)
- `void swap` (priority\_queue &other)

### 6.819.1 Detailed Description

`template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>`

`class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >`

A priority queue composed of one specific heap policy.

#### Template Parameters

|               |                                                       |
|---------------|-------------------------------------------------------|
| <i>_Tv</i>    | Value type.                                           |
| <i>Cmp_Fn</i> | Comparison functor.                                   |
| <i>Tag</i>    | Instantiating data structure type, see container_tag. |
| <i>_Alloc</i> | Allocator type.                                       |

Base is dispatched at compile time via Tag, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

### 6.819.2 Constructor & Destructor Documentation

#### [priority\\_queue\(\)](#) [1/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
 const cmp_fn & r_cmp_fn) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

#### [priority\\_queue\(\)](#) [2/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
 It first_it,
 It last_it) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

**priority\_queue()** [3/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
 It first_it,
 It last_it,
 const cmp_fn & r_cmp_fn) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_cmp\_fn will be copied by the cmp\_fn object of the container object. The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

**6.820 std::priority\_queue< \_Tp, \_Sequence, \_Compare > Class Template Reference**

```
#include <stl_queue.h>
```

**Public Types**

- typedef \_Sequence::const\_reference **const\_reference**
- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Sequence::value\_type **value\_type**

**Public Member Functions**

- template<typename \_Seq = \_Sequence, typename \_Requires = typename enable\_if<\_\_and<\_is\_default\_constructible<\_Compare>, is\_↵\_default\_constructible<\_Seq>>::value>::type>  
constexpr [priority\\_queue](#) ()
- template<typename \_InputIterator, typename \_Alloc, typename = std::\_RequireInputIter<\_InputIterator>, typename \_Requires = \_Uses<↵\_Alloc>>  
constexpr **priority\_queue** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_alloc)
- template<typename \_InputIterator, typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
constexpr **priority\_queue** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_x, \_Sequence &&\_\_s, const \_Alloc &\_\_alloc)
- template<typename \_InputIterator, typename \_Alloc, typename = std::\_RequireInputIter<\_InputIterator>, typename \_Requires = \_Uses<↵\_Alloc>>  
constexpr **priority\_queue** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_x, const \_Alloc &\_\_alloc)
- template<typename \_InputIterator, typename \_Alloc, typename = std::\_RequireInputIter<\_InputIterator>, typename \_Requires = \_Uses<↵\_Alloc>>  
constexpr **priority\_queue** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_x, const \_Sequence &&\_\_s, const \_Alloc &\_\_alloc)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
constexpr **priority\_queue** (const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
constexpr **priority\_queue** (const \_Compare &\_\_x, \_Sequence &&\_\_c, const \_Alloc &\_\_a)
- constexpr **priority\_queue** (const \_Compare &\_\_x, \_Sequence &&\_\_s=\_Sequence())
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
constexpr **priority\_queue** (const \_Compare &\_\_x, const \_Alloc &\_\_a)

- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`constexpr priority_queue (const _Compare &__x, const _Sequence &__c, const _Alloc &__a)`
- `constexpr priority_queue (const _Compare &__x, const _Sequence &__s)`
- `priority_queue (const priority_queue &)=default`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`constexpr priority_queue (const priority_queue &__q, const _Alloc &__a)`
- `constexpr priority_queue (priority_queue &&__q) noexcept(__and_< is\_nothrow\_move\_constructible< __↵  
Sequence >, is\_nothrow\_move\_constructible< _Compare > >::value)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`constexpr priority_queue (priority_queue &&__q, const _Alloc &__a)`
- `template<typename... _Args>`  
`constexpr void emplace (_Args &&... __args)`
- `constexpr bool empty () const`
- `priority\_queue & operator= (const priority\_queue &)=default`
- `constexpr priority\_queue & operator= (priority\_queue &&__q) noexcept(__and_< is\_nothrow\_move\_assignable< __↵  
_Sequence >, is\_nothrow\_move\_assignable< _Compare > >::value)`
- `constexpr void pop ()`
- `constexpr void push (const value_type &__x)`
- `constexpr void push (value_type &&__x)`
- `constexpr size_type size () const`
- `constexpr void swap (priority\_queue &__pq) noexcept(__and_< //c++1z or gnu++11 __is_nothrow_swappable< __↵  
_Sequence >, __is_nothrow_swappable< _Compare > >::value)`
- `constexpr const_reference top () const`
  
- `template<typename _InputIterator, typename = std::::RequireInputIter<_InputIterator>>`  
`constexpr priority\_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare())`
- `template<typename _InputIterator, typename = std::::RequireInputIter<_InputIterator>>`  
`constexpr priority\_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence  
&__s)`
- `template<typename _InputIterator, typename = std::::RequireInputIter<_InputIterator>>`  
`constexpr priority\_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, _Sequence &&__s)`

### Protected Attributes

- `_Sequence` `c`
- `_Compare` `comp`

### 6.820.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename __↵
Sequence::value_type>>
class std::priority_queue< _Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents.

### Template Parameters

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                                              |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .                     |
| <code>_Compare</code>  | Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> . |



This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

#### Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

### 6.820.2 Constructor & Destructor Documentation

#### `priority_queue()` [1/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and<is_default_constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue () [inline], [constexpr]
Default constructor creates no elements.
```

#### `priority_queue()` [2/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __x = _Compare()) [inline], [constexpr]
Builds a queue from a range.
```

#### Parameters

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

**priority\_queue()** [3/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __x,
 const _Sequence & __s) [inline], [constexpr]
```

Builds a queue from a range.

**Parameters**

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

**priority\_queue()** [4/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __x,
 _Sequence && __s) [inline], [constexpr]
```

Builds a queue from a range.

**Parameters**

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

References [std::move\(\)](#).

### 6.820.3 Member Function Documentation

#### empty()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
```

```
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty () const [inline], [nodiscard],
[constexpr]
```

Returns true if the queue is empty.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

#### pop()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
```

```
void std::priority_queue< _Tp, _Sequence, _Compare >::pop () [inline], [constexpr]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

#### push()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
```

```
void std::priority_queue< _Tp, _Sequence, _Compare >::push (
 const value_type & __x) [inline], [constexpr]
```

Add data to the queue.

#### Parameters

|                    |                   |
|--------------------|-------------------|
| <a href="#">_↵</a> | Data to be added. |
| <a href="#">_X</a> |                   |

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

#### size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
```

```
size_type std::priority_queue< _Tp, _Sequence, _Compare >::size () const [inline], [nodiscard],
[constexpr]
```

Returns the number of elements in the queue.

#### top()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
```

```
const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top () const [inline], [nodiscard],
[constexpr]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Referenced by [\\_\\_gnu\\_parallel::multiseq\\_partition\(\)](#), and [\\_\\_gnu\\_parallel::multiseq\\_selection\(\)](#).

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

## 6.821 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::priority\_queue\_tag:



### 6.821.1 Detailed Description

Basic priority-queue.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.822 \_\_gnu\_pbds::detail::probe\_fn\_base<\_Alloc> Class Template Reference

```
#include <probe_fn_base.hpp>
```

### 6.822.1 Detailed Description

```
template<typename _Alloc>
```

```
class __gnu_pbds::detail::probe_fn_base<_Alloc>
```

Probe functor base.

The documentation for this class was generated from the following file:

- [probe\\_fn\\_base.hpp](#)

## 6.823 \_\_gnu\_cxx::project1st<\_Arg1, \_Arg2> Struct Template Reference

```
#include <functional>
```

### Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Arg1 [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

**Public Member Functions**

- `_Arg1 operator()` (const `_Arg1` &`__x`, const `_Arg2` &) const

**6.823.1 Detailed Description**

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st<_Arg1, _Arg2>
```

An [SGI extension](#) .

**6.823.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Arg1>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

**result\_type**

```
typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Arg1>::result_type [inherited]
result_type is the return type
```

**second\_argument\_type**

```
typedef _Arg2 std::binary_function<_Arg1, _Arg2, _Arg1>::second_argument_type [inherited]
second_argument_type is the type of the second argument
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

**6.824 `__gnu_cxx::project2nd<_Arg1, _Arg2>` Struct Template Reference**

```
#include <functional>
```

**Public Types**

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Arg2` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

**Public Member Functions**

- `_Arg2 operator()` (const `_Arg1` &, const `_Arg2` &`__y`) const

**6.824.1 Detailed Description**

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd<_Arg1, _Arg2>
```

An [SGI extension](#) .

**6.824.2 Member Typedef Documentation****first\_argument\_type**

```
typedef _Arg1 std::binary_function<_Arg1, _Arg2, _Arg2>::first_argument_type [inherited]
first_argument_type is the type of the first argument
```

### result\_type

typedef \_Arg2 [std::binary\\_function](#)< \_Arg1, \_Arg2, \_Arg2 >::result\_type [inherited]  
result\_type is the return type

### second\_argument\_type

typedef \_Arg2 [std::binary\\_function](#)< \_Arg1, \_Arg2, \_Arg2 >::second\_argument\_type [inherited]  
second\_argument\_type is the type of the second argument  
The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 6.825 std::promise< \_Res > Class Template Reference

#include <future>

### Public Member Functions

- [template](#)<typename \_Allocator>  
    **promise** (allocator\_arg\_t, const \_Allocator &\_\_a)
- **promise** (const promise &)=delete
- **promise** (promise &&\_\_rhs) noexcept
- [future](#)< \_Res > **get\_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&\_\_rhs) noexcept
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** (\_Res &&\_\_r)
- void **set\_value** (const \_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &&\_\_r)
- void **set\_value\_at\_thread\_exit** (const \_Res &\_\_r)
- void **swap** (promise &\_\_rhs) noexcept

### Friends

- [template](#)<typename, typename>  
    struct \_State::Setter

#### 6.825.1 Detailed Description

**template**<typename \_Res>  
**class** std::promise< \_Res >

Primary template for promise.

The documentation for this class was generated from the following file:

- [future](#)

## 6.826 std::promise< \_Res & > Class Template Reference

#include <future>

**Public Member Functions**

- `template<typename _Allocator>`  
**promise** (allocator\_arg\_t, const \_Allocator &\_\_a)
- **promise** (const promise &)=delete
- **promise** (promise &&\_\_rhs) noexcept
- `future< _Res & >` **get\_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&\_\_rhs) noexcept
- void **set\_exception** (exception\_ptr \_\_p)
- void **set\_exception\_at\_thread\_exit** (exception\_ptr \_\_p)
- void **set\_value** (\_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &\_\_r)
- void **swap** (promise &\_\_rhs) noexcept

**Friends**

- `template<typename, typename>`  
struct \_State::\_Setter

**6.826.1 Detailed Description**

`template<typename _Res>`  
**class** `std::promise< _Res & >`

Partial specialization for `promise<R>`.

The documentation for this class was generated from the following file:

- `future`

**6.827 std::promise< void > Class Reference**

```
#include <future>
```

**Public Member Functions**

- `template<typename _Allocator>`  
**promise** (allocator\_arg\_t, const \_Allocator &\_\_a)
- **promise** (const promise &)=delete
- **promise** (promise &&\_\_rhs) noexcept
- `future< void >` **get\_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&\_\_rhs) noexcept
- void **set\_exception** (exception\_ptr \_\_p)
- void **set\_exception\_at\_thread\_exit** (exception\_ptr \_\_p)
- void **set\_value** ()
- void **set\_value\_at\_thread\_exit** ()
- void **swap** (promise &\_\_rhs) noexcept

**Friends**

- `template<typename, typename>`  
struct \_State::\_Setter

### 6.827.1 Detailed Description

Explicit specialization for `promise<void>`.

The documentation for this class was generated from the following file:

- [future](#)

## 6.828 `std::experimental::fundamentals_v2::propagate_const<_Tp>` Class Template Reference

```
#include <propagate_const>
```

### Public Types

- using `element_type`

### Public Member Functions

- `template<typename _Up, typename enable\_if< __and_< is\_constructible< _Tp, _Up && >, __not_< is\_convertible< _Up &&, _Tp > >, __not_< __is_propagate_const< typename decay< _Up >::type > >::value, bool >::type = false>`  
`constexpr propagate_const (_Up &&__u)`
- `template<typename _Up, typename enable\_if< __and_< is\_constructible< _Tp, _Up && >, is\_convertible< _Up &&, _Tp >, __not_< __is_propagate_const< typename decay< _Up >::type > >::value, bool >::type = true>`  
`constexpr propagate_const (_Up &&__u)`
- `propagate_const (const propagate_const &__p)=delete`
- `constexpr propagate_const (propagate_const &&__p)=default`
- `template<typename _Up, typename enable\_if< __and_< is\_constructible< _Tp, _Up && >, __not_< is\_convertible< _Up &&, _Tp > >::value, bool >::type = false>`  
`constexpr propagate_const (propagate_const< _Up > &&__pu)`
- `template<typename _Up, typename enable\_if< __and_< is\_constructible< _Tp, _Up && >, is\_convertible< _Up &&, _Tp > >::value, bool >::type = true>`  
`constexpr propagate_const (propagate_const< _Up > &&__pu)`
- `constexpr element_type * get ()`
- `constexpr const element_type * get () const`
- `constexpr operator bool () const`
- `constexpr element_type & operator* ()`
- `constexpr const element_type & operator* () const`
- `constexpr element_type * operator-> ()`
- `constexpr const element_type * operator-> () const`
- `template<typename _Up, typename = typename enable\_if< __and_< is\_convertible< _Up&&, _Tp>, __not_< __is_propagate_const< type-name decay<_Up>::type>> >::value>::type>`  
`constexpr propagate_const & operator= (_Up &&__u)`
- `propagate_const & operator= (const propagate_const &__p)=delete`
- `constexpr propagate_const & operator= (propagate_const &&__p)=default`
- `template<typename _Up, typename = typename enable\_if< is\_convertible< _Up&&, _Tp>::value>::type>`  
`constexpr propagate_const & operator= (propagate_const< _Up > &&__pu)`
- `constexpr void swap (propagate_const &__pt) noexcept(__is_nothrow_swappable< _Tp >::value)`

### Friends

- `template<typename _Up>`  
`constexpr const _Up & get_underlying (const propagate_const< _Up > &__pt) noexcept`
- `template<typename _Up>`  
`constexpr _Up & get_underlying (propagate_const< _Up > &__pt) noexcept`



### 6.828.1 Detailed Description

```
template<typename _Tp>
class std::experimental::fundamentals_v2::propagate_const< _Tp >
```

Const-propagating wrapper.

The documentation for this class was generated from the following file:

- [propagate\\_const](#)

## 6.829 `__gnu_pbds::quadratic_probe_fn< Size_Type >` Class Template Reference

```
#include <hash_policy.hpp>
```

### Public Types

- typedef `Size_Type` **size\_type**

### Public Member Functions

- void **swap** ([quadratic\\_probe\\_fn< Size\\_Type >](#) &other)

### Protected Member Functions

- `size_type` [operator\(\)](#) (`size_type` i) const

### 6.829.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

### 6.829.2 Member Function Documentation

#### **operator()**

```
template<typename Size_Type>
quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 6.830 `std::queue< _Tp, _Sequence >` Class Template Reference

```
#include <stl_queue.h>
```

### Public Types

- typedef `_Sequence::const_reference` **const\_reference**
- typedef `_Sequence` **container\_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size\_type**
- typedef `_Sequence::value_type` **value\_type**

## Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type> queue ()`
- `queue (_Sequence && __c)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>> queue (_Sequence && __c, const _Alloc & __a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>> queue (const _Alloc & __a)`
- `queue (const _Sequence & __c)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>> queue (const _Sequence & __c, const _Alloc & __a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>> queue (const queue & __q, const _Alloc & __a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>> queue (queue && __q, const _Alloc & __a)`
- reference `back ()`
- const\_reference `back () const`
- `template<typename... _Args> decltype(auto) emplace (_Args &&... __args)`
- bool `empty () const`
- reference `front ()`
- const\_reference `front () const`
- void `pop ()`
- void `push (const value_type & __x)`
- void `push (value_type && __x)`
- size\_type `size () const`
- void `swap (queue & __q) noexcept(__is_nothrow_swappable<_Sequence>::value)`

## Protected Attributes

- `_Sequence` [c](#)

## Friends

- `template<typename _Tp1, typename _Seq1> bool operator< (const queue<_Tp1, _Seq1> &, const queue<_Tp1, _Seq1> &)`
- `template<typename _Tp1, three_way_comparable _Seq1> compare_three_way_result_t<_Seq1> operator<=> (const queue<_Tp1, _Seq1> &, const queue<_Tp1, _Seq1> &)`
- `template<typename _Tp1, typename _Seq1> bool operator== (const queue<_Tp1, _Seq1> &, const queue<_Tp1, _Seq1> &)`

### 6.830.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue<_Tp, _Sequence>
```

A standard container giving FIFO behavior.

## Template Parameters

|                  |                  |
|------------------|------------------|
| <code>_Tp</code> | Type of element. |
|------------------|------------------|

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |
|------------------------|--------------------------------------------------------------------------|

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

## 6.830.2 Constructor & Destructor Documentation

### `queue()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↵
_Seq>::value>::type>
std::queue<_Tp, _Sequence>::queue () [inline]
```

Default constructor creates no elements.

References [c](#).

## 6.830.3 Member Function Documentation

### `back()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::back () [inline], [nodiscard]
```

Returns a read/write reference to the data at the last element of the queue.

References [c](#).

### `back()` [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::back () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the last element of the queue.

References [c](#).

### `empty()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue<_Tp, _Sequence>::empty () const [inline], [nodiscard]
```

Returns true if the queue is empty.

References [c](#).

### `front()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::front () [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the queue.

References [c](#).

**front()** [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::front () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

References [c](#).

**pop()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop () [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

References [c](#).

**push()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (
 const value_type & __x) [inline]
```

Add data to the end of the queue.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

References [c](#).

**size()**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const [inline], [nodiscard]
```

Returns the number of elements in the queue.

References [c](#).

**6.830.4 Member Data Documentation****c**

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequene std::queue< _Tp, _Sequence >::c [protected]
```

`c` is the underlying container.

Referenced by [queue\(\)](#), [back\(\)](#), [back\(\)](#), [empty\(\)](#), [front\(\)](#), [front\(\)](#), [std::operator<\(\)](#), [std::operator==\(\)](#), [pop\(\)](#), [push\(\)](#), and [size\(\)](#).

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

**6.831 \_\_gnu\_parallel::quicksort\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::quicksort_tag`:



### Public Member Functions

- `quicksort_tag` (`_ThreadIndex` \_\_num\_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` \_\_num\_threads)

#### 6.831.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

#### 6.831.2 Member Function Documentation

##### `__get_num_threads()`

`_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline], [inherited]

Find out desired number of threads.

##### Returns

Desired number of threads.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, and `__gnu_parallel::__parallel_sort()`.

##### `set_num_threads()`

void `__gnu_parallel::parallel_tag::set_num_threads` (  
     `_ThreadIndex` \_\_num\_threads) [inline], [inherited]

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 6.832 `std::random_access_iterator_tag` Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for `std::random_access_iterator_tag`:



#### 6.832.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations. The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 6.833 `__gnu_cxx::random_condition` Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::random_condition`:



## Classes

- struct [always\\_adjustor](#)
- struct [group\\_adjustor](#)
- struct [never\\_adjustor](#)

## Public Member Functions

- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### 6.833.1 Detailed Description

Base class for random probability control and throw.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.834 std::random\_device Class Reference

```
#include <random>
```

## Public Types

- typedef unsigned int [result\\_type](#)

## Public Member Functions

- **random\_device** (const random\_device &)=delete
- **random\_device** (const [std::string](#) &\_\_token)
- double **entropy** () const noexcept
- [result\\_type](#) **operator**() ()
- void **operator=** (const random\_device &)=delete

## Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

### 6.834.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Since

C++11

### 6.834.2 Member Typedef Documentation

#### result\_type

```
typedef unsigned int std::random_device::result_type
```

The type of the generated random value.

The documentation for this class was generated from the following file:

- [random.h](#)

## 6.835 std::range\_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for `std::range_error`:



### Public Member Functions

- **range\_error** (const char \*)
- **range\_error** (const range\_error &)=default
- **range\_error** (const [string](#) &\_\_arg)
- **range\_error** (range\_error &&)=default
- range\_error & **operator=** (const range\_error &)=default
- range\_error & **operator=** (range\_error &&)=default
- virtual const char \* [what](#) () const noexcept

### 6.835.1 Detailed Description

Thrown to indicate range errors in internal computations.



### 6.835.2 Member Function Documentation

#### `what()`

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 6.836 `__gnu_pbds::range_invalidation_guarantee` Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



#### 6.836.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 6.837 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`:



#### 6.837.1 Detailed Description

**template<typename Key, typename Hash\_Fn, typename \_Alloc, typename Comb\_Hash\_Fn, bool Store\_Hash>**  
**class** `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`

Primary template.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

### 6.838 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

#### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference) const
- void **swap** (ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, false > &)

### 6.838.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
```

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 6.839 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void **notify\_resized** (size\_type)
- comp\_hash **operator()** (key\_const\_reference) const
- comp\_hash **operator()** (key\_const\_reference, size\_type) const
- void **swap** (ranged\_hash\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Hash\_Fn, true > &)

### 6.839.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 6.840 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Comb\_Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Hash\_Fn &)
- void **swap** (ranged\_hash\_fn< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, false > &)

#### 6.840.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 6.841 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

```
#include <ranged_hash_fn.hpp>
```

### Protected Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_hash\_fn** (size\_type)
- **ranged\_hash\_fn** (size\_type, const Comb\_Hash\_Fn &)
- **ranged\_hash\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Hash\_Fn &)
- void **swap** (ranged\_hash\_fn< Key, [null\\_type](#), \_Alloc, Comb\_Hash\_Fn, true > &)

#### 6.841.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 6.842 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:



### 6.842.1 Detailed Description

`template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>`

`class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`

Primary template.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 6.843 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef Probe\_Fn **probe\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

### Protected Member Functions

- **ranged\_probe\_fn** (size\_type)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference) const
- size\_type **operator()** (key\_const\_reference, size\_type, size\_type) const
- void **swap** (ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, false > &)

#### 6.843.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1

The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

### 6.844 \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true > Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

#### Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef [std::pair](#)< size\_type, size\_type > **comp\_hash**
- typedef Hash\_Fn **hash\_fn\_base**
- typedef [rebind\\_traits](#)< \_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef Probe\_Fn **probe\_fn\_base**
- typedef \_Alloc::size\_type **size\_type**

#### Protected Member Functions

- **ranged\_probe\_fn** (size\_type)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &)
- **ranged\_probe\_fn** (size\_type, const Hash\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- void **notify\_resized** (size\_type)
- comp\_hash **operator()** (key\_const\_reference) const
- size\_type **operator()** (key\_const\_reference, size\_type) const
- size\_type **operator()** (key\_const\_reference, size\_type, size\_type) const
- void **swap** (ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, true > &)

#### 6.844.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
```

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 6.845 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

```
#include <ranged_probe_fn.hpp>
```

### Protected Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

### Protected Member Functions

- **ranged\_probe\_fn** (`size_type` size)
- **ranged\_probe\_fn** (`size_type`, const `Comb_Probe_Fn` &r\_comb\_probe\_fn)
- **ranged\_probe\_fn** (`size_type`, const `null_type` &, const `Comb_Probe_Fn` &r\_comb\_probe\_fn, const `null_type` &)
- void **swap** (`ranged_probe_fn` &other)

#### 6.845.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
```

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 6.846 `std::rank< _Tp >` Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for `std::rank< _Tp >`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr std::size\_t **value**

#### 6.846.1 Detailed Description

```
template<typename _Tp>
struct std::rank<_Tp>
```

rank

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 6.847 std::ratio<\_Num, \_Den > Struct Template Reference

```
#include <ratio>
```

#### Public Types

- typedef [ratio](#)< num, den > **type**

#### Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

#### 6.847.1 Detailed Description

```
template<intmax_t _Num, intmax_t _Den = 1>
struct std::ratio<_Num, _Den>
```

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax\_t. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

The documentation for this struct was generated from the following file:

- [ratio](#)

### 6.848 std::ratio\_equal<\_R1, \_R2 > Struct Template Reference

```
#include <ratio>
```



Inheritance diagram for `std::ratio_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 6.848.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_equal<_R1, _R2>
```

ratio\_equal

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 6.849 `std::ratio_greater<_R1, _R2>` Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_greater< _R1, _R2 >`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 6.849.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater< _R1, _R2 >
```

ratio\_greater

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 6.850 std::ratio\_greater\_equal< \_R1, \_R2 > Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_greater_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 6.850.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater_equal<_R1, _R2>
```

`ratio_greater_equal`

The documentation for this struct was generated from the following file:

- [ratio](#)

### 6.851 `std::ratio_less<_R1, _R2>` Struct Template Reference

```
#include <ratio>
```

#### 6.851.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_less<_R1, _R2>
```

`ratio_less`

The documentation for this struct was generated from the following file:

- [ratio](#)

### 6.852 `std::ratio_less_equal<_R1, _R2>` Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_less_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 6.852.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_less_equal<_R1, _R2>
```

`ratio_less_equal`

The documentation for this struct was generated from the following file:

- [ratio](#)

### 6.853 `std::ratio_not_equal<_R1, _R2>` Struct Template Reference

```
#include <ratio>
```

Inheritance diagram for `std::ratio_not_equal<_R1, _R2>`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 6.853.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_not_equal<_R1, _R2>
```

`ratio_not_equal`

The documentation for this struct was generated from the following file:

- [ratio](#)

## 6.854 `std::raw_storage_iterator<_OutputIterator, _Tp>` Class Template Reference

```
#include <stl_raw_storage_iter.h>
```

Inheritance diagram for `std::raw_storage_iterator<_OutputIterator, _Tp>`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) `iterator_category`
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **raw\_storage\_iterator** (`_OutputIterator __x`)
- `_OutputIterator` **base** () const
- `raw_storage_iterator` & **operator\*** ()
- `raw_storage_iterator` & **operator++** ()
- `raw_storage_iterator` **operator++** (int)
- `raw_storage_iterator` & **operator=** (`_Tp &&__element`)
- `raw_storage_iterator` & **operator=** (const `_Tp &__element`)

### Protected Attributes

- `_OutputIterator` **\_M\_iter**

#### 6.854.1 Detailed Description

```
template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator<_OutputIterator, _Tp>
```

This iterator class lets algorithms store their results into uninitialized memory.

#### 6.854.2 Member Typedef Documentation

##### `difference_type`

typedef void `std::iterator< output\_iterator\_tag, void, void, void, void >::difference_type` [inherited]  
 Distance between iterators is represented as this type.

**iterator\_category**

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↵
iterator_category [inherited]
```

One of the [tag types](#).

**pointer**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

**reference**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value\_type.

**value\_type**

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
```

The type "pointed to" by the iterator.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

## 6.855 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

```
#include <rb_tree>
```

**Public Types**

- `template<typename _Iter>`  
  using **\_\_same\_value\_type**
- `typedef std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Node_traits::_Const_iterator const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Node_traits::_Iterator iterator`
- `typedef _Key key_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _Val value_type`

**Public Member Functions**

- **rb\_tree** (const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **bool \_\_rb\_verify** () const
- `template<typename _Iterator>`  
  void **\_M\_assign\_equal** (\_Iterator, \_Iterator)

- `template<typename _Iterator>`  
`void _M_assign_unique (_Iterator, _Iterator)`
- `template<typename... _Args>`  
`iterator _M_emplace_equal (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator _M_emplace_here (bool __place_left, _Base_ptr __node, _Args &&... __args)`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_equal (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_unique (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`pair< iterator, bool > _M_emplace_unique (_Args &&... __args)`
- `template<typename _Kt>`  
`size_type _M_erase_tr (const _Kt &__x)`
- `size_type _M_erase_unique (const key_type &__x)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_equal_pos (const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_equal_pos (const_iterator __pos, const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_unique_pos (const_iterator __pos, const key_type &__k)`
- `template<typename _Kt>`  
`pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_unique_pos_tr (const_iterator, const _Kt &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_unique_pos (const key_type &__k)`
- `template<typename _Kt>`  
`pair< _Base_ptr, _Base_ptr > _M_get_insert_unique_pos_tr (const _Kt &__k)`
- `const _Node_allocator & _M_get_Node_allocator () const noexcept`
- `_Node_allocator & _M_get_Node_allocator () noexcept`
- `template<typename _Arg>`  
`iterator _M_insert_equal (_Arg &&__x)`
- `template<typename _Arg>`  
`iterator _M_insert_equal_ (const_iterator __pos, _Arg &&__x)`
- `template<typename _Arg, typename _NodeGen>`  
`iterator _M_insert_equal_ (const_iterator __pos, _Arg &&__x, _NodeGen &)`
- `template<typename _InputIterator>`  
`__enable_if_t<!_same_value_type< _InputIterator >::value > _M_insert_range_equal (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`__enable_if_t<!_same_value_type< _InputIterator >::value > _M_insert_range_equal (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`__enable_if_t<!_same_value_type< _InputIterator >::value > _M_insert_range_unique (_InputIterator __↵ first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`__enable_if_t<!_same_value_type< _InputIterator >::value > _M_insert_range_unique (_InputIterator __↵ first, _InputIterator __last)`
- `template<typename _Arg>`  
`pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > _M_insert_unique (_Arg &&__v)`
- `template<typename _Arg>`  
`pair< iterator, bool > _M_insert_unique (_Arg &&__x)`
- `template<typename _Arg>`  
`iterator _M_insert_unique_ (const_iterator __pos, _Arg &&__x)`
- `template<typename _Arg, typename _NodeGen>`  
`iterator _M_insert_unique_ (const_iterator __pos, _Arg &&__x, _NodeGen &)`



- `const_iterator` **begin** () `const noexcept`
- `iterator` **begin** () `noexcept`
- `void` **clear** () `noexcept`
- `size_type` **count** (const `key_type` &\_\_k) `const`
- `bool` **empty** () `const noexcept`
- `const_iterator` **end** () `const noexcept`
- `iterator` **end** () `noexcept`
- `pair< iterator, iterator >` **equal\_range** (const `key_type` &\_\_k)
- `pair< const_iterator, const_iterator >` **equal\_range** (const `key_type` &\_\_k) `const`
- `size_type` **erase** (const `key_type` &\_\_x)
- `iterator` **erase** (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator` **erase** (const `iterator` \_\_position)
- `iterator` **erase** (`iterator` \_\_position)
- `iterator` **find** (const `key_type` &\_\_k)
- `const_iterator` **find** (const `key_type` &\_\_k) `const`
- `allocator_type` **get\_allocator** () `const noexcept`
- `_Compare` **key\_comp** () `const`
- `iterator` **lower\_bound** (const `key_type` &\_\_k)
- `const_iterator` **lower\_bound** (const `key_type` &\_\_k) `const`
- `size_type` **max\_size** () `const noexcept`
- `const_reverse_iterator` **rbegin** () `const noexcept`
- `reverse_iterator` **rbegin** () `noexcept`
- `const_reverse_iterator` **rend** () `const noexcept`
- `reverse_iterator` **rend** () `noexcept`
- `size_type` **size** () `const noexcept`
- `void` **swap** (`_Rb_tree` &\_\_t) `noexcept( /*conditional */ )`
- `iterator` **upper\_bound** (const `key_type` &\_\_k)
- `const_iterator` **upper\_bound** (const `key_type` &\_\_k) `const`

### Protected Types

- `typedef _Node_traits::_Base_ptr` **\_Base\_ptr**
- `typedef _Node_traits::_Header_t` **\_Header\_t**
- `typedef _Node_traits::_Node_ptr` **\_Node\_ptr**

### Protected Member Functions

- `_Base_ptr` **\_M\_begin** () `const noexcept`
- `_Node_ptr` **\_M\_begin\_node** () `const noexcept`
- `template<bool _MoveValue, typename _NodeGen>`  
`_Node_ptr` **\_M\_clone\_node** (`_Node_ptr` \_\_x, `_NodeGen` &\_\_node\_gen)
- `template<typename... _Args>`  
`void` **\_M\_construct\_node** (`_Node_ptr` \_\_node, `_Args` &&... \_\_args)
- `template<typename... _Args>`  
`_Node_ptr` **\_M\_create\_node** (`_Args` &&... \_\_args)
- `void` **\_M\_destroy\_node** (`_Node_ptr` \_\_p) `noexcept`
- `void` **\_M\_drop\_node** (`_Node_ptr` \_\_p) `noexcept`
- `_Base_ptr` **\_M\_end** () `const noexcept`
- `_Node_ptr` **\_M\_get\_node** ()
- `template<typename _Key1, typename _Key2>`  
`bool` **\_M\_key\_compare** (const `_Key1` &\_\_k1, const `_Key2` &\_\_k2) `const`
- `_Base_ptr` **\_M\_leftmost** () `const noexcept`

- `_Base_ptr & _M_leftmost () noexcept`
- `void _M_put_node (_Node_ptr __p) noexcept`
- `_Base_ptr _M_rightmost () const noexcept`
- `_Base_ptr & _M_rightmost () noexcept`
- `_Base_ptr _M_root () const noexcept`
- `_Base_ptr & _M_root () noexcept`

#### Static Protected Member Functions

- `static const _Key & _S_key (_Base_ptr __x)`
- `static const _Key & _S_key (_Node_ptr __x)`
- `static const _Key & _S_key (const _Node &__node)`
- `static _Base_ptr _S_left (_Base_ptr __x) noexcept`
- `static _Node_ptr _S_left (_Node_ptr __x)`
- `static _Base_ptr _S_right (_Base_ptr __x) noexcept`
- `static _Node_ptr _S_right (_Node_ptr __x) noexcept`

#### Protected Attributes

- `_Rb_tree_impl< _Compare > _M_impl`

#### 6.855.1 Detailed Description

`template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = std::allocator<↵  
_Value>>`

`struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵  
documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html)

The documentation for this struct was generated from the following file:

- [rb\\_tree](#)

#### 6.856 `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <rb_tree_.hpp>
```

#### Public Types

- `typedef _Alloc allocator_type`
- `typedef Cmp_Fn cmp_fn`
- `typedef std::pair< size_type, size_type > comp_hash`
- `typedef base_type::const_iterator const_iterator`
- `typedef base_type::const_pointer const_pointer`
- `typedef base_type::const_reference const_reference`
- `typedef base_type::const_reverse_iterator const_reverse_iterator`
- `typedef rb\_tree\_tag container_category`
- `typedef _Alloc::difference_type difference_type`
- `typedef base_type::iterator iterator`
- `typedef base_type::key_const_pointer key_const_pointer`

- `typedef base_type::key_const_reference` **key\_const\_reference**
- `typedef base_type::key_pointer` **key\_pointer**
- `typedef base_type::key_reference` **key\_reference**
- `typedef base_type::key_type` **key\_type**
- `typedef base_type::mapped_const_pointer` **mapped\_const\_pointer**
- `typedef base_type::mapped_const_reference` **mapped\_const\_reference**
- `typedef base_type::mapped_pointer` **mapped\_pointer**
- `typedef base_type::mapped_reference` **mapped\_reference**
- `typedef base_type::mapped_type` **mapped\_type**
- `typedef __nothrowcopy::indicator` **no\_throw\_indicator**
- `typedef traits_type::node_const_iterator` **node\_const\_iterator**
- `typedef traits_type::node_iterator` **node\_iterator**
- `typedef base_type::node_update` **node\_update**
- `typedef base_type::const_iterator` **point\_const\_iterator**
- `typedef base_type::point_iterator` **point\_iterator**
- `typedef base_type::pointer` **pointer**
- `typedef base_type::reference` **reference**
- `typedef base_type::reverse_iterator` **reverse\_iterator**
- `typedef _Alloc::size_type` **size\_type**
- `typedef integral_constant< int, Store_Hash >` **store\_extra**
- `typedef stored\_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- `typedef base_type::value_type` **value\_type**

### Public Member Functions

- **rb\_tree\_map** (const Cmp\_Fn &)
- **rb\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **rb\_tree\_map** (const rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It>  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference)
- void **join** (rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_iterator [node\\_begin](#) ()

- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_end](#) ()
- node\_const\_iterator [node\\_end](#) () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_alloc\_traits::value\_type **node**
- typedef node\_alloc\_traits::allocator\_type **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_subtree\_size** (node\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

#### 6.856.1 Detailed Description

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵  
Alloc>`

`class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

#### 6.856.2 Member Function Documentation

##### **node\_begin()** [1/2]

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵  
Alloc>`

`bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵  
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin  
( ) [inline], [inherited]`

Returns a `node_iterator` corresponding to the node at the root of the tree.

##### **node\_begin()** [2/2]

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵  
Alloc>`

`bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵  
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵  
begin ( ) const [inline], [inherited]`

Returns a const `node_iterator` corresponding to the node at the root of the tree.

##### **node\_end()** [1/2]

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵  
Alloc>`

`bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵  
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( )  
[inline], [inherited]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

##### **node\_end()** [2/2]

`template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↵  
Alloc>`

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
end () const [inline], [inherited]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [rb\\_tree\\_.hpp](#)

## 6.857 `__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

```
#include <node.hpp>
```

### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference metadata\_const\_reference
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::reference metadata\_reference
- typedef Metadata metadata\_type
- typedef [rebind\\_traits](#)< \_Alloc, [rb\\_tree\\_node\\_](#) >::pointer node\_pointer
- typedef [rebind\\_traits](#)< \_Alloc, [rb\\_tree\\_node\\_](#) >::size\_type size\_type
- typedef Value\_Type value\_type

### Public Member Functions

- metadata\_reference **get\_metadata** ()
- metadata\_const\_reference **get\_metadata** () const
- bool **special** () const

### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_red**
- size\_type **m\_subtree\_size**
- value\_type **m\_value**

#### 6.857.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for Red-Black trees.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/node.hpp](#)

## 6.858 `__gnu_pbds::rb_tree_tag` Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::rb\_tree\_tag:



### 6.858.1 Detailed Description

Red-black tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.859 \_\_gnu\_pbds::detail::rc< \_Node, \_Alloc > Class Template Reference

```
#include <rc.hpp>
```

### Public Types

- typedef entry\_const\_pointer **const\_iterator**
- typedef node\_pointer **entry**

### Public Member Functions

- **rc** (const rc &)
- const const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const\_iterator **end** () const

- void **pop** ()
- void **push** (entry)
- size\_type **size** () const
- void **swap** (rc &)
- node\_pointer **top** () const

### 6.859.1 Detailed Description

```
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >
```

Redundant binary counter.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

## 6.860 \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

```
#include <rc_binomial_heap_.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



### Public Types

- typedef base\_type::allocator\_type **allocator\_type**
- typedef base\_type::cmp\_fn **cmp\_fn**
- typedef base\_type::const\_iterator **const\_iterator**
- typedef base\_type::const\_pointer **const\_pointer**
- typedef base\_type::const\_reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef base\_type::point\_const\_iterator **point\_const\_iterator**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef base\_type::pointer **pointer**
- typedef base\_type::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef Value\_Type **value\_type**

### Public Member Functions

- **rc\_binomial\_heap** (const Cmp\_Fn &)
- **rc\_binomial\_heap** (const rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- iterator **begin** ()
- void **clear** ()



- bool **empty** () const
- iterator **end** ()
- void **erase** (point\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- void **join** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **join** (rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- template<typename Pred>  
void **split** (Pred, rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- void **swap** (rc\_binomial\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef base\_type::node **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef `std::pair`< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** (binomial\_heap\_base< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

### 6.860.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Redundant-counter binomial heap.

The documentation for this class was generated from the following file:

- [rc\\_binomial\\_heap.hpp](#)

### 6.861 \_\_gnu\_pbds::rc\_binomial\_heap\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::rc\_binomial\_heap\_tag:



#### 6.861.1 Detailed Description

Redundant-counter binomial-heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 6.862 std::simd::rebind<\_Tp, \_Mp, \_Traits> Struct Template Reference

```
#include <simd_mask.h>
```

#### Public Types

- using `type`

### 6.862.1 Detailed Description

```
template<__vectorizable _Tp, __simd_mask_type _Mp, _ArchTraits _Traits>
struct std::simd::rebind<_Tp, _Mp, _Traits>
```

As above, except for `basic_mask`.

The documentation for this struct was generated from the following file:

- `simd_mask.h`

## 6.863 `std::simd::rebind<_Tp, _Vp, _Traits>` Struct Template Reference

```
#include <simd_mask.h>
```

### Public Types

- using `type`

### 6.863.1 Detailed Description

```
template<__vectorizable _Tp, __simd_vec_type _Vp, _ArchTraits _Traits>
struct std::simd::rebind<_Tp, _Vp, _Traits>
```

Computes a member type `basic_vec<_Tp, Abi>`, where `Abi` is chosen such that the number of elements is equal to `_Vp::size()` and features of the ABI tag (such as the internal representation of masks, or storage order of complex components) are preserved.

The documentation for this struct was generated from the following file:

- `simd_mask.h`

## 6.864 `__gnu_pbds::detail::rebind_traits<_Alloc, T>` Struct Template Reference

```
#include <types_traits.hpp>
```

### Public Types

- using `const_reference`
- using `reference`

### 6.864.1 Detailed Description

```
template<typename _Alloc, typename T>
struct __gnu_pbds::detail::rebind_traits<_Alloc, T>
```

Consistent API for accessing allocator-related types.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.865 `std::filesystem::recursive_directory_iterator` Class Reference

```
#include <filesystem>
```

## Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [input\\_iterator\\_tag](#) **iterator\_category**
- typedef const [directory\\_entry](#) \* **pointer**
- typedef const [directory\\_entry](#) & **reference**
- typedef [directory\\_entry](#) **value\_type**

## Public Member Functions

- **recursive\_directory\_iterator** (const [path](#) &\_\_p)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [directory\\_options](#) \_\_options, [error\\_code](#) &\_\_ec)
- **recursive\_directory\_iterator** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- **recursive\_directory\_iterator** (const recursive\_directory\_iterator &)=default
- **recursive\_directory\_iterator** (recursive\_directory\_iterator &&)=default
- int **depth** () const noexcept
- void **disable\_recursion\_pending** () noexcept
- recursive\_directory\_iterator & **increment** ([error\\_code](#) &\_\_ec)
- const [directory\\_entry](#) & **operator\*** () const noexcept
- recursive\_directory\_iterator & **operator++** ()
- \_\_directory\_iterator\_proxy **operator++** (int)
- const [directory\\_entry](#) \* **operator->** () const noexcept
- recursive\_directory\_iterator & **operator=** (const recursive\_directory\_iterator &\_\_rhs) noexcept
- recursive\_directory\_iterator & **operator=** (recursive\_directory\_iterator &&\_\_rhs) noexcept
- bool **operator==** ([default\\_sentinel\\_t](#)) const noexcept
- [directory\\_options](#) **options** () const noexcept
- void **pop** ()
- void **pop** ([error\\_code](#) &)
- bool **recursion\_pending** () const noexcept

## Friends

- uintmax\_t **filesystem::remove\_all** (const [path](#) &)
- uintmax\_t **filesystem::remove\_all** (const [path](#) &, [error\\_code](#) &)
- bool **operator==** (const recursive\_directory\_iterator &\_\_lhs, const recursive\_directory\_iterator &\_\_rhs) noexcept

### 6.865.1 Detailed Description

Iterator type for recursively traversing a directory hierarchy.

Since

C++17

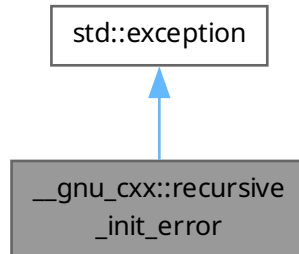
The documentation for this class was generated from the following file:

- [bits/fs\\_dir.h](#)

## 6.866 `__gnu_cxx::recursive_init_error` Class Reference

```
#include <cxxabi.h>
```

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



### Public Member Functions

- virtual constexpr const char \* [what](#) () const noexcept

#### 6.866.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

#### 6.866.2 Member Function Documentation

##### `what()`

```
virtual constexpr const char * std::exception::what () const [inline], [constexpr], [virtual],
[noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad\\_alloc](#), [std::bad\\_cast](#), [std::bad\\_exception](#), [std::bad\\_function\\_call](#), [std::bad\\_typeid](#), [std::bad\\_weak\\_ptr](#), [std::experimental::filesystem::v1::filesystem\\_error](#), [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#), [std::filesystem::filesystem\\_error](#), [std::future\\_error](#), [std::ios\\_base::failure](#), [std::logic\\_error](#), and [std::runtime\\_error](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 6.867 `std::recursive_mutex` Class Reference

```
#include <mutex>
```

### Public Types

- typedef `__native_type` \* **native\_handle\_type**

### Public Member Functions

- **recursive\_mutex** (const recursive\_mutex &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- recursive\_mutex & **operator=** (const recursive\_mutex &)=delete
- bool **try\_lock** () noexcept
- void **unlock** ()

#### 6.867.1 Detailed Description

The standard recursive mutex type.

A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

### 6.868 std::recursive\_timed\_mutex Class Reference

```
#include <mutex>
```

#### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

#### Public Member Functions

- **recursive\_timed\_mutex** (const recursive\_timed\_mutex &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- recursive\_timed\_mutex & **operator=** (const recursive\_timed\_mutex &)=delete
- bool **try\_lock** () noexcept
- template<class \_Rep, class \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<class \_Clock, class \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

#### Friends

- class **\_\_timed\_mutex\_impl**< recursive\_timed\_mutex >

#### 6.868.1 Detailed Description

The standard recursive timed mutex type.

A recursive mutex that supports a timeout when trying to acquire the lock. A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

## 6.869 `std::bitset<_Nb>::reference` Class Reference

```
#include <bitset>
```

### Public Member Functions

- **reference** (const reference &)=default
- constexpr reference & **flip** () noexcept
- constexpr **operator bool** () const noexcept
- constexpr reference & **operator=** (bool \_\_x) noexcept
- constexpr reference & **operator=** (const reference &\_\_j) noexcept
- constexpr bool **operator~** () const noexcept

### Friends

- class **bitset**

#### 6.869.1 Detailed Description

```
template<size_t _Nb>
```

```
class std::bitset<_Nb>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

The documentation for this class was generated from the following file:

- [bitset](#)

## 6.870 `std::tr2::dynamic_bitset<_WordT, _Alloc>::reference` Class Reference

```
#include <dynamic_bitset>
```

### Public Member Functions

- **reference** (dynamic\_bitset &\_\_b, size\_type \_\_pos) noexcept
- reference & **flip** () noexcept
- **operator bool** () const noexcept
- reference & **operator=** (bool \_\_x) noexcept
- reference & **operator=** (const reference &\_\_j) noexcept
- bool **operator~** () const noexcept

### Friends

- class **dynamic\_bitset**

### 6.870.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

### 6.871 std::reference\_wrapper< \_Tp > Class Template Reference

```
#include <refwrap.h>
```

#### Public Types

- typedef `_Tp type`

#### Public Member Functions

- template<typename `_Up`, typename = `__not_same<_Up>`, typename = `decltype(reference_wrapper::_S_fun(std::declval<_Up>()))`> constexpr **reference\_wrapper** (`_Up &&__uref`) noexcept(noexcept(reference\_wrapper::\_S\_fun([std::declval](#)< `_Up` >())))
- **reference\_wrapper** (const reference\_wrapper &)=default
- constexpr `_Tp & get` () const noexcept
- constexpr **operator \_Tp &** () const noexcept
- template<typename... `_Args`> constexpr `__invoke_result< _Tp &, _Args... >::type operator()` (`_Args &&... __args`) const noexcept(`__is_`↔`nothrow_invocable< _Tp &, _Args... >::value`)
- reference\_wrapper & **operator=** (const reference\_wrapper &)=default

#### Friends

- constexpr auto **operator<=**> (reference\_wrapper `__x`, const `_Tp &__y`)
- constexpr auto **operator<=**> (reference\_wrapper `__x`, reference\_wrapper `__y`)
- constexpr auto **operator<=**> (reference\_wrapper `__x`, reference\_wrapper< const `_Tp` > `__y`)
- constexpr bool **operator==** (reference\_wrapper `__x`, const `_Tp &__y`)
- constexpr bool **operator==** (reference\_wrapper `__x`, reference\_wrapper `__y`)
- constexpr bool **operator==** (reference\_wrapper `__x`, reference\_wrapper< const `_Tp` > `__y`)

#### Related Symbols

(Note that these are not member symbols.)

- template<typename `_Tp`> constexpr reference\_wrapper< `_Tp` > [ref](#) (`_Tp &__t`) noexcept
- template<typename `_Tp`> constexpr reference\_wrapper< const `_Tp` > [cref](#) (const `_Tp &__t`) noexcept
- template<typename `_Tp`> constexpr reference\_wrapper< `_Tp` > [ref](#) (reference\_wrapper< `_Tp` > `__t`) noexcept
- template<typename `_Tp`> constexpr reference\_wrapper< const `_Tp` > [cref](#) (reference\_wrapper< `_Tp` > `__t`) noexcept



### 6.871.1 Detailed Description

```
template<typename _Tp>
class std::reference_wrapper< _Tp >
```

Primary class template for reference\_wrapper.

### 6.871.2 Friends And Related Symbol Documentation

#### cref() [1/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
 const _Tp & __t) [related]
```

Denotes a const reference should be taken to a variable.

#### cref() [2/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
 reference_wrapper< _Tp > __t) [related]
```

std::cref overload to prevent wrapping a reference\_wrapper

#### ref() [1/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
 _Tp & __t) [related]
```

Denotes a reference should be taken to a variable.

#### ref() [2/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
 reference_wrapper< _Tp > __t) [related]
```

std::ref overload to prevent wrapping a reference\_wrapper

The documentation for this class was generated from the following files:

- [type\\_traits](#)
- [refwrap.h](#)

## 6.872 std::regex\_error Class Reference

```
#include <regex>
```

Inheritance diagram for `std::regex_error`:



#### Public Member Functions

- [regex\\_error](#) ([error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type](#) code () const noexcept
- virtual const char \* [what](#) () const noexcept

#### 6.872.1 Detailed Description

A regular expression exception class.  
The regular expression library throws objects of this class on error.

Since

C++11

#### 6.872.2 Constructor & Destructor Documentation

##### **regex\_error()**

```
std::regex_error::regex_error (
 error_type __ecode) [explicit]
```

Constructs a `regex_error` object.

##### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__ecode</code> | the regex error code. |
|----------------------|-----------------------|

References [regex\\_error\(\)](#).

Referenced by [regex\\_error\(\)](#).

### 6.872.3 Member Function Documentation

#### `code()`

```
regex_constants::error_type std::regex_error::code () const [inline], [noexcept]
```

Gets the regex error code.

#### Returns

the regex error code.

References [code\(\)](#).

Referenced by [code\(\)](#).

#### `what()`

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

## 6.873 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

```
#include <regex>
```

### Public Types

- `typedef std::ptrdiff_t difference_type`
- `typedef std::forward\_iterator\_tag iterator_category`
- `typedef std::input\_iterator\_tag iterator_concept`
- `typedef const value_type * pointer`
- `typedef const value_type & reference`
- `typedef basic\_regex< \_Ch\_type, \_Rx\_traits > regex_type`
- `typedef match\_results< \_Bi\_iter > value_type`

### Public Member Functions

- `regex\_iterator ()=default`
- `regex\_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex\_constants::match\_flag\_type __m= regex\_constants::match\_default)`
- `regex_iterator (_Bi_iter, _Bi_iter, const regex_type &&, regex\_constants::match\_flag\_type=regex\_constants::match\_default)=deleted`
- `regex\_iterator (const regex\_iterator &)=default`
- `const value_type & operator\* () const noexcept`
- `regex\_iterator & operator++ ()`
- `regex\_iterator operator++ (int)`
- `const value_type * operator-> () const noexcept`
- `regex\_iterator & operator= (const regex\_iterator &)=default`
- `bool operator== (const regex\_iterator &) const noexcept`
- `bool operator== (default\_sentinel\_t) const noexcept`

### 6.873.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Since

C++11

### 6.873.2 Constructor & Destructor Documentation

#### `regex_iterator()` [1/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator () [default]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Referenced by `std::regex_iterator< const char * >::regex_iterator()`, and `operator==( )`.

#### `regex_iterator()` [2/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a `regex_iterator`...

#### Parameters

|                   |                                                    |
|-------------------|----------------------------------------------------|
| <code>__a</code>  | [IN] The start of a text range to search.          |
| <code>__b</code>  | [IN] One-past-the-end of the text range to search. |
| <code>__re</code> | [IN] The regular expression to match.              |
| <code>__m</code>  | [IN] Policy flags for match rules.                 |

#### `regex_iterator()` [3/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
 const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > &) [default]
```

Copy constructs a `regex_iterator`.

### 6.873.3 Member Function Documentation

#### `operator*()`

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator* () const
[inline], [noexcept]
```

Dereferences a `regex_iterator`.

#### `operator++()` [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ ()
```

Increments a `regex_iterator`.

References `std::regex_constants::match_continuous`, `std::regex_constants::match_not_null`, `std::regex_constants::match_prev_avail`, and `std::regex_search()`.

#### `operator++()` [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++ (
 int) [inline]
```

Postincrements a `regex_iterator`.

#### `operator->()`

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator-> () const
[inline], [noexcept]
```

Selects a `regex_iterator` member.

#### `operator=()`

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator & std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator= (
 const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> &) [default]
```

Copy assigns one `regex_iterator` to another.

#### `operator==()`

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator== (
 const regex_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs) const [noexcept]
```

Tests the equivalence of two `regex iterators`.

References `regex_iterator()`.

The documentation for this class was generated from the following files:

- `regex.h`
- `regex.tcc`

## 6.874 `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>` Class Template Reference

```
#include <regex>
```

### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `std::input_iterator_tag` **iterator\_concept**
- typedef `const value_type *` **pointer**
- typedef `const value_type &` **reference**
- typedef `basic_regex<_Ch_type, _Rx_traits>` **regex\_type**
- typedef `sub_match<_Bi_iter>` **value\_type**

### Public Member Functions

- `regex_token_iterator()`
- `template<std::size_t Nm> regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[Nm]↔, regex_constants::match_flag_type __m=regex_constants::match_default)`
- `regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector<int> &__↔submatches, regex_constants::match_flag_type __m=regex_constants::match_default)`
- `regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, initializer_list<int> __submatches, regex_constants::match_flag_type __m=regex_constants::match_default)`
- `regex_token_iterator(_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)`
- `template<std::size_t Nm> regex_token_iterator(_Bi_iter, _Bi_iter, const regex_type &&, const int(&)[Nm], regex_constants::match_flag_type=regex_consta`
- `regex_token_iterator(_Bi_iter, _Bi_iter, const regex_type &&, const std::vector<int> &, regex_constants::match_flag_type=regex`
- `regex_token_iterator(_Bi_iter, _Bi_iter, const regex_type &&, initializer_list<int>, regex_constants::match_flag_type=regex_con`
- `regex_token_iterator(_Bi_iter, _Bi_iter, const regex_type &&, int=0, regex_constants::match_flag_type=regex_constants::match_c`
- `regex_token_iterator(const regex_token_iterator &__rhs)`
- `const value_type & operator*() const`
- `regex_token_iterator & operator++()`
- `regex_token_iterator operator++(int)`
- `const value_type * operator->() const`
- `regex_token_iterator & operator= (const regex_token_iterator &__rhs)`
- `bool operator== (const regex_token_iterator &__rhs) const`
- `bool operator== (default_sentinel_t) const noexcept`

### 6.874.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a `std::sub_match` object.

Since

C++11

### 6.874.2 Constructor & Destructor Documentation

#### regex\_token\_iterator() [1/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator () [inline]
```

Default constructs a regex\_token\_iterator.

A default-constructed regex\_token\_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Referenced by [operator++\(\)](#), [operator=\(\)](#), and [operator==\(\)](#).

#### regex\_token\_iterator() [2/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 int __submatch = 0,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a regex\_token\_iterator...

#### Parameters

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__a</code>        | [IN] The start of the text to search.                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>__b</code>        | [IN] One-past-the-end of the text to search.                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>__re</code>       | [IN] The regular expression to search for.                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>__submatch</code> | [IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> <li>• -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> <li>• 0 the entire string matching the subexpression is returned for each match within the text.</li> <li>• &gt;0 enumerates only the indicated subexpression from a match within the text.</li> </ul> |
| <code>__m</code>        | [IN] Policy flags for match rules.                                                                                                                                                                                                                                                                                                                                                                                                 |

#### regex\_token\_iterator() [3/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 const std::vector< int > & __submatches,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a regex\_token\_iterator...

**Parameters**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**regex\_token\_iterator() [4/6]**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 initializer_list< int > __submatches,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a `regex_token_iterator`...

**Parameters**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**regex\_token\_iterator() [5/6]**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
template<std::size_t _Nm>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 const int (&) __submatches[_Nm],
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a `regex_token_iterator`...



**Parameters**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

**regex\_token\_iterator() [6/6]**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) [inline]
```

Copy constructs a regex\_token\_iterator.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|

**6.874.3 Member Function Documentation****operator\*()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
const value_type & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()
const [inline]
```

Dereferences a regex\_token\_iterator.

**operator++() [1/2]**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()
```

Increments a regex\_token\_iterator.

References [regex\\_token\\_iterator\(\)](#).

**operator++() [2/2]**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
```

```
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
 int) [inline]
```

Postincrements a regex\_token\_iterator.

**operator->()**

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()
const [inline]
```

Selects a `regex_token_iterator` member.

**operator=()**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Assigns a `regex_token_iterator` to another.

**Parameters**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__rhs</code> | [IN] A <code>regex_token_iterator</code> to copy. |
|--------------------|---------------------------------------------------|

References [regex\\_token\\_iterator\(\)](#).

**operator==()**

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const
```

Compares a `regex_token_iterator` to another for equality.

References [regex\\_token\\_iterator\(\)](#).

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

**6.875 std::regex\_traits< \_Ch\_type > Class Template Reference**

```
#include <regex>
```

**Public Types**

- typedef `_RegexMask` **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef [std::locale](#) **locale\_type**
- typedef [std::basic\\_string](#)< `char_type` > **string\_type**

**Public Member Functions**

- [regex\\_traits](#) ()
- `locale_type` [getloc](#) () const
- `locale_type` [imbue](#) (`locale_type` \_\_loc)
- bool [isctype](#) (`_Ch_type` \_\_c, `char_class_type` \_\_f) const
- template<typename `_Fwd_iter`>  
`char_class_type` [lookup\\_classname](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last, bool \_\_icase=false) const
- template<typename `_Fwd_iter`>  
`string_type` [lookup\\_collatename](#) (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const

- template<typename \_Fwd\_iter>  
string\_type [transform](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- template<typename \_Fwd\_iter>  
string\_type [transform\\_primary](#) (\_Fwd\_iter \_\_first, \_Fwd\_iter \_\_last) const
- char\_type [translate](#) (char\_type \_\_c) const
- char\_type [translate\\_nocase](#) (char\_type \_\_c) const
- int [value](#) (\_Ch\_type \_\_ch, int \_\_radix) const

### Static Public Member Functions

- static std::size\_t [length](#) (const char\_type \*\_\_p)

### Protected Attributes

- locale\_type [\\_M\\_locale](#)

### 6.875.1 Detailed Description

**template<typename \_Ch\_type>**  
**class std::regex\_traits<\_Ch\_type>**

Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Since

C++11

### 6.875.2 Constructor & Destructor Documentation

#### regex\_traits()

```
template<typename _Ch_type>
std::regex_traits<_Ch_type>::regex_traits () [inline]
```

Constructs a default traits object.

Referenced by [value\(\)](#).

### 6.875.3 Member Function Documentation

#### getloc()

```
template<typename _Ch_type>
locale_type std::regex_traits<_Ch_type>::getloc () const [inline]
```

Gets a copy of the current locale in use by the regex\_traits object.

#### imbue()

```
template<typename _Ch_type>
locale_type std::regex_traits<_Ch_type>::imbue (
 locale_type __loc) [inline]
```

Imbues the regex\_traits object with a copy of a new locale.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

**Returns**

a copy of the previous locale in use by the `regex_traits` object.

**Note**

Calling `imbue` with a different locale than the one currently in use invalidates all cached data held by `*this`.

**isctype()**

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::isctype (
 _Ch_type __c,
 char_class_type __f) const
```

Determines if `c` is a member of an identified class.

**Parameters**

|                  |                                                                 |
|------------------|-----------------------------------------------------------------|
| <code>__c</code> | a character.                                                    |
| <code>__f</code> | a class type (as returned from <code>lookup_classname</code> ). |

**Returns**

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

**Exceptions**

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a ctype facet. |
|----------------------------|----------------------------------------------------|

References `std::use_facet()`.

**length()**

```
template<typename _Ch_type>
std::size_t std::regex_traits< _Ch_type >::length (
 const char_type * __p) [inline], [static]
```

Gives the length of a C-style string starting at `__p`.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__p</code> | a pointer to the start of a character sequence. |
|------------------|-------------------------------------------------|

**Returns**

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

**lookup\_classname()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits<_Ch_type>::char_class_type std::regex_traits<_Ch_type>::lookup_classname (
 _Fwd_iter __first,
 _Fwd_iter __last,
 bool __icase = false) const
```

Maps one or more characters to a named character classification.

**Parameters**

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>__first</code> | beginning of the character sequence.         |
| <code>__last</code>  | one-past-the-end of the character sequence.  |
| <code>__icase</code> | ignores the case of the classification name. |

**Returns**

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s
- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower

- `print`
- `punct`
- `space`
- `upper`
- `xdigit`

References [std::use\\_facet\(\)](#).

### **lookup\_collatename()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::string_type std::regex_traits< _Ch_type >::lookup_collatename (
 _Fwd_iter __first,
 _Fwd_iter __last) const
```

Gets a collation element by name.

#### **Parameters**

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>__first</code> | beginning of the collation element name.        |
| <code>__last</code>  | one-past-the-end of the collation element name. |

#### **Returns**

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range `[__first, __last)`. Returns an empty string if the character sequence is not a valid collating element.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::empty\(\)](#), and [std::use\\_facet\(\)](#).

### **transform()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform (
 _Fwd_iter __first,
 _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence.

#### **Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Returns a sort key for the character sequence designated by the iterator range `[F1, F2)` such that if the character sequence `[G1, G2)` sorts before the character sequence `[H1, H2)` then `v.transform(G1, G2) < v.transform(H1, H2)`. What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

**Returns**

a locale-specific sort key equivalent to the input range.

**Exceptions**

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a collate facet. |
|----------------------------|------------------------------------------------------|

**transform\_primary()**

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits<_Ch_type>::transform_primary (
 _Fwd_iter __first,
 _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence, independent of case.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

**Effects:** if `typeid(use_facet<collate<_Ch_type>>(getloc())) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function correctly.

**translate()**

```
template<typename _Ch_type>
char_type std::regex_traits<_Ch_type>::translate (
 char_type __c) const [inline]
```

Performs the identity translation.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

**Returns**

`__c`.

**translate\_nocase()**

```
template<typename _Ch_type>
char_type std::regex_traits<_Ch_type>::translate_nocase (
 char_type __c) const [inline]
```

Translates a character into a case-insensitive equivalent.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

**Returns**

the locale-specific lower-case equivalent of `__c`.

**Exceptions**

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <code>std::bad_cast</code> | if the imbued locale does not support the ctype facet. |
|----------------------------|--------------------------------------------------------|

**value()**

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
 _Ch_type __ch,
 int __radix) const
```

Converts a digit to an int.

**Parameters**

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| <code>__ch</code>    | a character representing a digit.                              |
| <code>__radix</code> | the radix if the numeric conversion (limited to 8, 10, or 16). |

**Returns**

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

References [regex\\_traits\(\)](#), and [std::use\\_facet\(\)](#).

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex.tcc](#)

**6.876 std::remove\_all\_extents< \_Tp > Struct Template Reference**

```
#include <type_traits>
```

**Public Types**

- using `type`

**6.876.1 Detailed Description**

```
template<typename _Tp>
struct std::remove_all_extents< _Tp >
```

`remove_all_extents`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 6.877 `std::remove_const<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

#### 6.877.1 Detailed Description

```
template<typename _Tp>
struct std::remove_const<_Tp>
```

`remove_const`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.878 `std::remove_cv<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

#### 6.878.1 Detailed Description

```
template<typename _Tp>
struct std::remove_cv<_Tp>
```

`remove_cv`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.879 `std::remove_extent<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### Public Types

- using `type`

#### 6.879.1 Detailed Description

```
template<typename _Tp>
struct std::remove_extent<_Tp>
```

`remove_extent`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.880 `std::remove_pointer<_Tp>` Struct Template Reference

```
#include <type_traits>
```

## Public Types

- using **type**

### 6.880.1 Detailed Description

```
template<typename _Tp>
struct std::remove_pointer< _Tp >
```

remove\_pointer

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.881 std::remove\_reference< \_Tp > Struct Template Reference

```
#include <type_traits>
```

## Public Types

- using **type**

### 6.881.1 Detailed Description

```
template<typename _Tp>
struct std::remove_reference< _Tp >
```

remove\_reference

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.882 std::remove\_volatile< \_Tp > Struct Template Reference

```
#include <type_traits>
```

## Public Types

- using **type**

### 6.882.1 Detailed Description

```
template<typename _Tp>
struct std::remove_volatile< _Tp >
```

remove\_volatile

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.883 \_\_gnu\_pbds::resize\_error Struct Reference

```
#include <exception.hpp>
```

Inheritance diagram for `__gnu_pbds::resize_error`:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 6.883.1 Detailed Description

A container cannot be resized.

#### 6.883.2 Member Function Documentation

##### `what()`

```
virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 6.884 `__gnu_pbds::detail::resize_policy<_Tp>` Class Template Reference

```
#include <resize_policy.hpp>
```

## Public Types

- typedef `_Tp` **size\_type**

## Public Member Functions

- **resize\_policy** (const `resize_policy` &other)
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (`size_type`) const
- void **notify\_arbitrary** (`size_type`)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- bool **resize\_needed\_for\_grow** (`size_type`) const
- bool **resize\_needed\_for\_shrink** (`size_type`) const
- bool **shrink\_needed** (`size_type`) const
- void **swap** (`resize_policy`< `_Tp` > &)

## Static Public Attributes

- static const `_Tp` **min\_size**

### 6.884.1 Detailed Description

```
template<typename _Tp>
class __gnu_pbds::detail::resize_policy< _Tp >
```

Resize policy for binary heap.

The documentation for this class was generated from the following file:

- [resize\\_policy.hpp](#)

## 6.885 `std::result_of< _Signature >` Struct Template Reference

### 6.885.1 Detailed Description

```
template<typename _Signature>
struct std::result_of< _Signature >
```

`result_of`

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.886 `std::reverse_iterator< _Iterator >` Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for `std::reverse_iterator<_Iterator>`:



### Public Types

- using **difference\_type**
- using **iterator\_category**
- using **iterator\_concept**
- typedef `_Iterator` **iterator\_type**
- typedef `__traits_type::pointer` **pointer**
- using **reference**
- using **value\_type**

### Public Member Functions

- constexpr `reverse_iterator` () noexcept(*/\*conditional \*/*)
- constexpr `reverse_iterator` (const `reverse_iterator` &\_\_x) noexcept(*/\*conditional \*/*)
- template<typename `_Iter`>  
constexpr `reverse_iterator` (const `reverse_iterator`< `_Iter` > &\_\_x) noexcept(*/\*conditional \*/*)
- constexpr `reverse_iterator` (`iterator_type` \_\_x) noexcept(*/\*conditional \*/*)
- constexpr `iterator_type` `base` () const noexcept(*/\*conditional \*/*)
- constexpr reference `operator*` () const
- constexpr `reverse_iterator` `operator+` (`difference_type` \_\_n) const
- constexpr `reverse_iterator` & `operator++` ()
- constexpr `reverse_iterator` `operator++` (int)
- constexpr `reverse_iterator` & `operator+=` (`difference_type` \_\_n)
- constexpr `reverse_iterator` `operator-` (`difference_type` \_\_n) const
- constexpr `reverse_iterator` & `operator--` ()
- constexpr `reverse_iterator` `operator--` (int)
- constexpr `reverse_iterator` & `operator-=` (`difference_type` \_\_n)
- constexpr pointer `operator->` () const
- `reverse_iterator` & `operator=` (const `reverse_iterator` &)=default
- template<typename `_Iter`>  
constexpr `reverse_iterator` & `operator=` (const `reverse_iterator`< `_Iter` > &\_\_x) noexcept(*/\*conditional \*/*)
- constexpr reference `operator[]` (`difference_type` \_\_n) const

## Protected Types

- typedef [iterator\\_traits](#)< \_Iterator > [\\_\\_traits\\_type](#)

## Protected Attributes

- \_Iterator **current**

## Friends

- constexpr [iter\\_rvalue\\_reference\\_t](#)< \_Iterator > **iter\_move** (const reverse\_iterator &\_\_i) noexcept(is\_nothrow\_↵ copy\_constructible\_v< \_Iterator > &&noexcept(ranges::iter\_move(--std::declval< \_Iterator & >())))
- template<indirectly\_swappable< \_Iterator > \_Iter2>  
constexpr void **iter\_swap** (const reverse\_iterator &\_\_x, const reverse\_iterator< \_Iter2 > &\_\_y) noexcept(is\_↵ nothrow\_copy\_constructible\_v< \_Iterator > &&is\_nothrow\_copy\_constructible\_v< \_Iter2 > &&noexcept(ranges::↵ iter\_swap(--std::declval< \_Iterator & >(), --std::declval< \_Iter2 & >())))

### 6.886.1 Detailed Description

**template<typename \_Iterator>**

**class std::reverse\_iterator< \_Iterator >**

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator *i* is established by the identity:

```
&*(reverse_iterator(i)) == &(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array.* [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

### 6.886.2 Constructor & Destructor Documentation

**reverse\_iterator()** [1/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator< _Iterator >::reverse_iterator () [inline], [constexpr], [noexcept]
```

The default constructor value-initializes member **current**. If it is a pointer, that means it is zero-initialized.

Referenced by [std::reverse\\_iterator< iterator >::operator+\(\)](#), and [std::reverse\\_iterator< iterator >::operator-\(\)](#).

**reverse\_iterator()** [2/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator< _Iterator >::reverse_iterator (
 iterator_type __x) [inline], [explicit], [constexpr], [noexcept]
```

This iterator will move in the opposite direction that *x* does.

**reverse\_iterator()** [3/4]

```
template<typename _Iterator>
```

```
std::reverse_iterator< _Iterator >::reverse_iterator (
 const reverse_iterator< _Iterator > & __x) [inline], [constexpr], [noexcept]
```

The copy constructor is normal.

**reverse\_iterator()** [4/4]

```
template<typename _Iterator>
template<typename _Iter>
std::reverse_iterator<_Iterator>::reverse_iterator (
 const reverse_iterator<_Iter> &__x) [inline], [constexpr], [noexcept]
```

A reverse\_iterator across other types can be copied if the underlying iterator can be converted to the type of current.

**6.886.3 Member Function Documentation****base()**

```
template<typename _Iterator>
iterator_type std::reverse_iterator<_Iterator>::base () const [inline], [nodiscard], [constexpr],
[noexcept]
```

**Returns**

current, the iterator used for underlying work.

Referenced by `std::operator==()`.

**operator\*()**

```
template<typename _Iterator>
reference std::reverse_iterator<_Iterator>::operator* () const [inline], [nodiscard], [constexpr]
```

**Returns**

A reference to the value at `--current`

This requires that `--current` is dereferenceable.

**Warning**

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

**operator+()**

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator<_Iterator>::operator+ (
 difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

A reverse\_iterator that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

**operator++()** [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator<_Iterator>::operator++ () [inline], [constexpr]
```

**Returns**

\*this

Decrements the underlying iterator.

**operator++()** [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
 int) [inline], [constexpr]
```

**Returns**

The original value of `*this`

Decrements the underlying iterator.

**operator+=()**

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator+= (
 difference_type __n) [inline], [constexpr]
```

**Returns**

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

**operator-()**

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator- (
 difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

**operator--()** [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-- () [inline], [constexpr]
```

**Returns**

`*this`

Increments the underlying iterator.

**operator--()** [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator-- (
 int) [inline], [constexpr]
```

**Returns**

A `reverse_iterator` with the previous value of `*this`

Increments the underlying iterator.



**operator-=( )**

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-= (
 difference_type __n) [inline], [constexpr]
```

**Returns**

\*this

Moves the underlying iterator forwards `__n` steps. The underlying iterator must be a Random Access Iterator.

**operator->()**

```
template<typename _Iterator>
pointer std::reverse_iterator< _Iterator >::operator-> () const [inline], [nodiscard], [constexpr]
```

**Returns**

A pointer to the value at `--current`

This requires that `--current` is dereferenceable.

**operator[]()**

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator[] (
 difference_type __n) const [inline], [nodiscard], [constexpr]
```

**Returns**

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

**6.887 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference**

```
#include <rope>
```

**Public Types**

- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` `__C`
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` `__F`
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` `__L`
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` `__S`
- typedef `__alloc_traits<_Alloc>::template rebind<__C>::other` `_CAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<_CharT>::other` `_DataAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__F>::other` `_FAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__L>::other` `_LAlloc`
- typedef `__alloc_traits<_Alloc>::template rebind<__S>::other` `_SAlloc`
- typedef `_Rope_const_iterator<_CharT, _Alloc>` `const_iterator`
- typedef `const _CharT *` `const_pointer`
- typedef `_CharT` `const_reference`
- typedef `std::reverse_iterator<const_iterator>` `const_reverse_iterator`
- typedef `std::ptrdiff_t` `difference_type`

- typedef `_Rope_iterator` < `_CharT`, `_Alloc` > **iterator**
- typedef `_Rope_char_ptr_proxy` < `_CharT`, `_Alloc` > **pointer**
- typedef `_Rope_char_ref_proxy` < `_CharT`, `_Alloc` > **reference**
- typedef `std::reverse_iterator` < `iterator` > **reverse\_iterator**
- typedef `std::size_t` **size\_type**
- typedef `_CharT` **value\_type**

## Public Member Functions

- **rope** (`_CharT` `__c`, `const allocator_type & __a=allocator_type()`)
- **rope** (`char_producer` < `_CharT` > `* __fn`, `size_type __len`, `bool __delete_fn`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `const _CharT * __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const _CharT * __s`, `size_type __len`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const allocator_type & __a=allocator_type()`)
- **rope** (`const const_iterator & __s`, `const const_iterator & __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const iterator & __s`, `const iterator & __e`, `const allocator_type & __a=allocator_type()`)
- **rope** (`const rope & __x`, `const allocator_type & __a=allocator_type()`)
- **rope** (`size_type __n`, `_CharT __c`, `const allocator_type & __a=allocator_type()`)
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `rope & append ()`
- `rope & append (_CharT __c)`
- `rope & append (const _CharT * __c_string)`
- `rope & append (const _CharT * __iter, size_type __n)`
- `rope & append (const _CharT * __s, const _CharT * __e)`
- `rope & append (const rope & __y)`
- `rope & append (const_iterator __s, const_iterator __e)`
- `rope & append (size_type __n, _CharT __c)`
- `void apply_to_pieces (size_type __begin, size_type __end, _Rope_char_consumer < _CharT > & __c) const`
- `_CharT at (size_type __pos) const`
- `_CharT back () const`
- `void balance ()`
- `const_iterator begin ()`
- `const_iterator begin () const`
- `const _CharT * c_str () const`
- `void clear ()`
- `int compare (const rope & __y) const`
- `const_iterator const_begin () const`
- `const_iterator const_end () const`
- `const_reverse_iterator const_rbegin () const`
- `const_reverse_iterator const_rend () const`
- `void copy (_CharT * __buffer) const`
- `size_type copy (size_type __pos, size_type __n, _CharT * __buffer) const`
- `void delete_c_str ()`
- `void dump ()`
- `bool empty () const`
- `const_iterator end ()`
- `const_iterator end () const`
- `iterator erase (const iterator & __p)`

- iterator **erase** (const iterator &\_\_p, const iterator &\_\_q)
- void **erase** (size\_type \_\_p, size\_type \_\_n)
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- \_CharT **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*c\_string)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const rope &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **insert** (size\_type \_\_p)
- void **insert** (size\_type \_\_p, \_CharT \_\_c)
- void **insert** (size\_type \_\_p, const \_CharT \*c\_string)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **insert** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **insert** (size\_type \_\_p, const rope &\_\_r)
- void **insert** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()
- reverse\_iterator **mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- reverse\_iterator **mutable\_rend** ()
- rope & **operator=** (const rope &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- const\_reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- const\_reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*c\_string)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const\_iterator \_\_i, const\_iterator \_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- void **replace** (size\_type \_\_p, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_type \_\_p, const rope &\_\_r)
- void **replace** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const rope &\_\_r)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- rope< \_CharT, \_Alloc > **substr** (const\_iterator \_\_start)
- rope **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- rope **substr** (iterator \_\_start) const
- rope **substr** (iterator \_\_start, iterator \_\_end) const
- rope **substr** (size\_type \_\_start, size\_type \_\_len=1) const
- void **swap** (rope &\_\_b)

### Static Public Member Functions

- static \_\_C \* **\_C\_allocate** (std::size\_t \_\_n)
- static void **\_C\_deallocate** (\_\_C \*\_\_p, std::size\_t \_\_n)
- static \_CharT \* **\_Data\_allocate** (std::size\_t \_\_n)
- static void **\_Data\_deallocate** (\_CharT \*\_\_p, std::size\_t \_\_n)
- static \_\_F \* **\_F\_allocate** (std::size\_t \_\_n)
- static void **\_F\_deallocate** (\_\_F \*\_\_p, std::size\_t \_\_n)
- static \_\_L \* **\_L\_allocate** (std::size\_t \_\_n)
- static void **\_L\_deallocate** (\_\_L \*\_\_p, std::size\_t \_\_n)
- static \_\_S \* **\_S\_allocate** (std::size\_t \_\_n)
- static void **\_S\_deallocate** (\_\_S \*\_\_p, std::size\_t \_\_n)

### Public Attributes

- \_RopeRep \* **\_M\_tree\_ptr**

### Static Public Attributes

- static const size\_type **npos**

### Protected Types

- enum { **\_S\_copy\_max** }
- typedef `_Rope_base<_CharT, _Alloc>` **\_Base**
- typedef `_CharT*` **\_Cstrptr**
- typedef `_Rope_RopeConcatenation<_CharT, _Alloc>` **\_RopeConcatenation**
- typedef `_Rope_RopeFunction<_CharT, _Alloc>` **\_RopeFunction**
- typedef `_Rope_RopeLeaf<_CharT, _Alloc>` **\_RopeLeaf**
- typedef `_Rope_RopeRep<_CharT, _Alloc>` **\_RopeRep**
- typedef `_Rope_RopeSubstring<_CharT, _Alloc>` **\_RopeSubstring**
- typedef `_Rope_self_destruct_ptr<_CharT, _Alloc>` **\_Self\_destruct\_ptr**
- typedef `_Base::allocator_type` **allocator\_type**

### Static Protected Member Functions

- static `size_type` **\_S\_allocated\_capacity** (`size_type` \_\_n)
- static `bool` **\_S\_apply\_to\_pieces** (`_Rope_char_consumer<_CharT>` &\_\_c, `const _RopeRep*` \_\_r, `size_type` \_\_begin, `size_type` \_\_end)
- static `_RopeRep*` **\_S\_concat** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right)
- static `_RopeRep*` **\_S\_concat\_char\_iter** (`_RopeRep*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen, `allocator_type` &\_\_a)
- static `_RopeRep*` **\_S\_destr\_concat\_char\_iter** (`_RopeRep*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen, `allocator_type` &\_\_a)
- static `_RopeLeaf*` **\_S\_destr\_leaf\_concat\_char\_iter** (`_RopeLeaf*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen)
- static `_CharT` **\_S\_fetch** (`_RopeRep*` \_\_r, `size_type` \_\_pos)
- static `_CharT*` **\_S\_fetch\_ptr** (`_RopeRep*` \_\_r, `size_type` \_\_pos)
- static `bool` **\_S\_is0** (`_CharT` \_\_c)
- static `_RopeLeaf*` **\_S\_leaf\_concat\_char\_iter** (`_RopeLeaf*` \_\_r, `const _CharT*` \_\_iter, `size_type` \_\_slen)
- static `_RopeConcatenation*` **\_S\_new\_RopeConcatenation** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right, `allocator_type` &\_\_a)
- static `_RopeFunction*` **\_S\_new\_RopeFunction** (`char_producer<_CharT>` \*\_\_f, `size_type` \_\_size, `bool` \_\_d, `allocator_type` &\_\_a)
- static `_RopeLeaf*` **\_S\_new\_RopeLeaf** (`_CharT*` \_\_s, `size_type` \_\_size, `allocator_type` &\_\_a)
- static `_RopeSubstring*` **\_S\_new\_RopeSubstring** (`_Rope_RopeRep<_CharT, _Alloc>` \*\_\_b, `size_type` \_\_s, `size_type` \_\_l, `allocator_type` &\_\_a)
- static `void` **\_S\_ref** (`_RopeRep*` \_\_t)
- static `_RopeLeaf*` **\_S\_RopeLeaf\_from\_unowned\_char\_ptr** (`const _CharT*` \_\_s, `size_type` \_\_size, `allocator_type` &\_\_a)
- static `size_type` **\_S\_rounded\_up\_size** (`size_type` \_\_n)
- static `_RopeRep*` **\_S\_substring** (`_RopeRep*` \_\_base, `size_type` \_\_start, `size_type` \_\_endp1)
- static `_RopeRep*` **\_S\_tree\_concat** (`_RopeRep*` \_\_left, `_RopeRep*` \_\_right)
- static `void` **\_S\_unref** (`_RopeRep*` \_\_t)
- static `_RopeRep*` **replace** (`_RopeRep*` \_\_old, `size_type` \_\_pos1, `size_type` \_\_pos2, `_RopeRep*` \_\_r)

### Static Protected Attributes

- static `_CharT` **\_S\_empty\_c\_str** [1]

## Friends

- class `_Rope_char_ptr_proxy<_CharT, _Alloc>`
- class `_Rope_char_ref_proxy<_CharT, _Alloc>`
- class `_Rope_const_iterator<_CharT, _Alloc>`
- class `_Rope_iterator<_CharT, _Alloc>`
- class `_Rope_iterator_base<_CharT, _Alloc>`
- struct `_Rope_RopeRep<_CharT, _Alloc>`
- struct `_Rope_RopeSubstring<_CharT, _Alloc>`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, _CharT2 __right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const _CharT2 *__right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const rope<_CharT2, _Alloc2> &__right)`

### 6.887.1 Detailed Description

```
template<class _CharT, class _Alloc>
class __gnu_cxx::rope<_CharT, _Alloc>
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html)

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

## 6.888 std::runtime\_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for `std::runtime_error`:



## Public Member Functions

- `runtime_error (const char *)`
- `runtime_error (const runtime_error &) noexcept`

- [runtime\\_error](#) (const [string](#) &\_\_arg)
- [runtime\\_error](#) (runtime\_error &&) noexcept
- [runtime\\_error](#) & **operator=** (const [runtime\\_error](#) &) noexcept
- [runtime\\_error](#) & **operator=** ([runtime\\_error](#) &&) noexcept
- virtual const char \* [what](#) () const noexcept

### 6.888.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

### 6.888.2 Constructor & Destructor Documentation

#### `runtime_error()`

```
std::runtime_error::runtime_error (
 const string & __arg) [explicit]
```

Takes a character string describing the error.

References [runtime\\_error\(\)](#).

Referenced by [runtime\\_error\(\)](#).

### 6.888.3 Member Function Documentation

#### `what()`

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept\\_except.h](#)

## 6.889 `__gnu_pbds::sample_probe_fn` Class Reference

```
#include <sample_probe_fn.hpp>
```

### Public Types

- typedef std::size\_t **size\_type**

### Public Member Functions

- [sample\\_probe\\_fn](#) ()
- [sample\\_probe\\_fn](#) (const [sample\\_probe\\_fn](#) &)
- void [swap](#) ([sample\\_probe\\_fn](#) &)

### Protected Member Functions

- [size\\_type](#) [operator\(\)](#) (key\_const\_reference r\_key, [size\\_type](#) i) const

### 6.889.1 Detailed Description

A sample probe policy.

### 6.889.2 Constructor & Destructor Documentation

#### sample\_probe\_fn() [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ()
```

Default constructor.

Referenced by [sample\\_probe\\_fn\(\)](#), and [swap\(\)](#).

#### sample\_probe\_fn() [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (
 const sample_probe_fn &)
```

Copy constructor.

References [sample\\_probe\\_fn\(\)](#).

### 6.889.3 Member Function Documentation

#### operator()()

```
size_type __gnu_pbds::sample_probe_fn::operator() (
 key_const_reference r_key,
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value of some key r\_key.

#### swap()

```
void __gnu_pbds::sample_probe_fn::swap (
 sample_probe_fn &) [inline]
```

Swaps content.

References [sample\\_probe\\_fn\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

## 6.890 \_\_gnu\_pbds::sample\_range\_hashing Class Reference

```
#include <sample_range_hashing.hpp>
```

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_range\\_hashing](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

### Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type](#) [operator\(\)](#) ([size\\_type](#)) const

### 6.890.1 Detailed Description

A sample range-hashing functor.



## 6.890.2 Member Typedef Documentation

### size\_type

typedef std::size\_t \_\_gnu\_pbds::sample\_range\_hashing::size\_type  
Size type.

## 6.890.3 Constructor & Destructor Documentation

### sample\_range\_hashing() [1/2]

\_\_gnu\_pbds::sample\_range\_hashing::sample\_range\_hashing ()  
Default constructor.  
Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

### sample\_range\_hashing() [2/2]

\_\_gnu\_pbds::sample\_range\_hashing::sample\_range\_hashing (  
const sample\_range\_hashing & other)  
Copy constructor.  
References [sample\\_range\\_hashing\(\)](#).

## 6.890.4 Member Function Documentation

### notify\_resized()

void \_\_gnu\_pbds::sample\_range\_hashing::notify\_resized (  
size\_type ) [protected]  
Notifies the policy object that the container's size has changed to argument's size.

### operator>()

size\_type \_\_gnu\_pbds::sample\_range\_hashing::operator() (  
size\_type ) const [inline], [protected]  
Transforms the \_\_hash value hash into a ranged-hash value.

### swap()

void \_\_gnu\_pbds::sample\_range\_hashing::swap (  
sample\_range\_hashing & other) [inline]  
Swaps content.  
References [sample\\_range\\_hashing\(\)](#).  
The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

## 6.891 \_\_gnu\_pbds::sample\_ranged\_hash\_fn Class Reference

#include <sample\_ranged\_hash\_fn.hpp>

### Public Types

- typedef std::size\_t **size\_type**

## Public Member Functions

- [sample\\_ranged\\_hash\\_fn](#) ()
- [sample\\_ranged\\_hash\\_fn](#) (const [sample\\_ranged\\_hash\\_fn](#) &)
- void [swap](#) ([sample\\_ranged\\_hash\\_fn](#) &)

## Protected Member Functions

- void [notify\\_resized](#) (size\_type)
- size\_type [operator\(\)](#) (key\_const\_reference) const

### 6.891.1 Detailed Description

A sample ranged-hash functor.

### 6.891.2 Constructor & Destructor Documentation

#### [sample\\_ranged\\_hash\\_fn](#)() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()
```

Default constructor.

Referenced by [sample\\_ranged\\_hash\\_fn](#)(), and [swap](#)().

#### [sample\\_ranged\\_hash\\_fn](#)() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (
 const sample_ranged_hash_fn &)
```

Copy constructor.

References [sample\\_ranged\\_hash\\_fn](#)().

### 6.891.3 Member Function Documentation

#### [notify\\_resized](#)()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (
 size_type) [protected]
```

Notifies the policy object that the container's `__size` has changed to `size`.

#### [operator](#)()()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (
 key_const_reference) const [inline], [protected]
```

Transforms `key_const_reference` into a position within the table.

#### [swap](#)()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (
 sample_ranged_hash_fn &) [inline]
```

Swaps content.

References [sample\\_ranged\\_hash\\_fn](#)().

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

## 6.892 \_\_gnu\_pbds::sample\_ranged\_probe\_fn Class Reference

```
#include <sample_ranged_probe_fn.hpp>
```

## Public Types

- typedef std::size\_t **size\_type**

## Public Member Functions

- **sample\_ranged\_probe\_fn** (const sample\_ranged\_probe\_fn &)
- void **swap** (sample\_ranged\_probe\_fn &)

## Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

### 6.892.1 Detailed Description

A sample ranged-probe functor.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

## 6.893 `__gnu_pbds::sample_resize_policy` Class Reference

```
#include <sample_resize_policy.hpp>
```

## Public Types

- typedef std::size\_t [size\\_type](#)

## Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void [swap](#) ([sample\\_resize\\_policy](#) &other)

## Protected Member Functions

- [size\\_type](#) [get\\_new\\_size](#) ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_e)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_e)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

### 6.893.1 Detailed Description

A sample resize policy.

### 6.893.2 Member Typedef Documentation

#### **size\_type**

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```

Size type.

### 6.893.3 Constructor & Destructor Documentation

#### **sample\_resize\_policy()**

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ()
```

Default constructor.  
Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

### 6.893.4 Member Function Documentation

#### **get\_new\_size()**

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (
 size_type size,
 size_type num_used_e) const [protected]
```

Queries what the new size should be.

#### **is\_resize\_needed()**

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

#### **notify\_cleared()**

```
void __gnu_pbds::sample_resize_policy::notify_cleared () [protected]
```

Notifies the table was cleared.

#### **notify\_erase\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### **notify\_erase\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

#### **notify\_erase\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

#### **notify\_erased()**

```
void __gnu_pbds::sample_resize_policy::notify_erased (
 size_type num_e) [inline], [protected]
```

Notifies an element was erased.

**notify\_find\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_find\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_inserted()**

```
void __gnu_pbds::sample_resize_policy::notify_inserted (
 size_type num_e) [inline], [protected]
```

Notifies an element was inserted.

**notify\_resized()**

```
void __gnu_pbds::sample_resize_policy::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized to new\_size.

**sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (
 const sample_resize_policy & other)
```

Copy constructor.

References [sample\\_resize\\_policy\(\)](#).

**swap()**

```
void __gnu_pbds::sample_resize_policy::swap (
 sample_resize_policy & other) [inline]
```

Swaps content.

References [sample\\_resize\\_policy\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 6.894 \_\_gnu\_pbds::sample\_resize\_trigger Class Reference

```
#include <sample_resize_trigger.hpp>
```

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_trigger](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_trigger](#) &)
- void [swap](#) ([sample\\_resize\\_trigger](#) &)

### Protected Member Functions

- bool [is\\_grow\\_needed](#) ([size\\_type](#) size, [size\\_type](#) num\_entries) const
- bool [is\\_resize\\_needed](#) () const
- void [notify\\_cleared](#) ()
- void [notify\\_erase\\_search\\_collision](#) ()
- void [notify\\_erase\\_search\\_end](#) ()
- void [notify\\_erase\\_search\\_start](#) ()
- void [notify\\_erased](#) ([size\\_type](#) num\_entries)
- void [notify\\_externally\\_resized](#) ([size\\_type](#) new\_size)
- void [notify\\_find\\_search\\_collision](#) ()
- void [notify\\_find\\_search\\_end](#) ()
- void [notify\\_find\\_search\\_start](#) ()
- void [notify\\_insert\\_search\\_collision](#) ()
- void [notify\\_insert\\_search\\_end](#) ()
- void [notify\\_insert\\_search\\_start](#) ()
- void [notify\\_inserted](#) ([size\\_type](#) num\_entries)
- void [notify\\_resized](#) ([size\\_type](#) new\_size)

#### 6.894.1 Detailed Description

A sample resize trigger policy.

#### 6.894.2 Member Typedef Documentation

##### **size\_type**

```
typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type
```

Size type.

#### 6.894.3 Constructor & Destructor Documentation

##### **sample\_resize\_trigger()**

```
__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()
```

Default constructor.

Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

#### 6.894.4 Member Function Documentation

##### is\_grow\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
 size_type size,
 size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed.

##### is\_resize\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

##### notify\_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared () [protected]
```

Notifies the table was cleared.

##### notify\_erase\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

##### notify\_erase\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

##### notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

##### notify\_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
 size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

##### notify\_externally\_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
 size_type new_size) [protected]
```

Notifies the table was resized externally.

##### notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

##### notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**notify\_inserted()**

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
 size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. the total number of entries in the table is num\_entries.

**notify\_resized()**

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

**sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (
 const sample_resize_trigger &)
```

Copy constructor.  
References [sample\\_resize\\_trigger\(\)](#).

**swap()**

```
void __gnu_pbds::sample_resize_trigger::swap (
 sample_resize_trigger &) [inline]
```

Swaps content.  
References [sample\\_resize\\_trigger\(\)](#).  
The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

**6.895 \_\_gnu\_pbds::sample\_size\_policy Class Reference**

```
#include <sample_size_policy.hpp>
```

**Public Types**

- typedef std::size\_t [size\\_type](#)



## Public Member Functions

- [sample\\_size\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_size\\_policy](#) &)
- void [swap](#) ([sample\\_size\\_policy](#) &other)

## Protected Member Functions

- [size\\_type](#) [get\\_nearest\\_larger\\_size](#) ([size\\_type](#) size) const
- [size\\_type](#) [get\\_nearest\\_smaller\\_size](#) ([size\\_type](#) size) const

### 6.895.1 Detailed Description

A sample size policy.

### 6.895.2 Member Typedef Documentation

#### `size_type`

typedef std::size\_t `__gnu_pbds::sample_size_policy::size_type`  
Size type.

### 6.895.3 Constructor & Destructor Documentation

#### `sample_size_policy()`

`__gnu_pbds::sample_size_policy::sample_size_policy` ()

Default constructor.

Referenced by [sample\\_range\\_hashing\(\)](#), and [swap\(\)](#).

### 6.895.4 Member Function Documentation

#### `get_nearest_larger_size()`

[size\\_type](#) `__gnu_pbds::sample_size_policy::get_nearest_larger_size` (  
    [size\\_type](#) size) const [inline], [protected]

Given a `__size` size, returns a `__size` that is larger.

#### `get_nearest_smaller_size()`

[size\\_type](#) `__gnu_pbds::sample_size_policy::get_nearest_smaller_size` (  
    [size\\_type](#) size) const [inline], [protected]

Given a `__size` size, returns a `__size` that is smaller.

#### `sample_range_hashing()`

`__gnu_pbds::sample_size_policy::sample_range_hashing` (  
    const [sample\\_size\\_policy](#) & )

Copy constructor.

References [sample\\_size\\_policy\(\)](#).

#### `swap()`

void `__gnu_pbds::sample_size_policy::swap` (  
    [sample\\_size\\_policy](#) & other) [inline]

Swaps content.

References [sample\\_size\\_policy\(\)](#).

The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

## 6.896 `__gnu_pbds::sample_tree_node_update`< `Const_Node_Iter`, `Node_Iter`, `Cmp_Fn`, `_Alloc` > Class Template Reference

```
#include <sample_tree_node_update.hpp>
```

### 6.896.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updatator.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

## 6.897 `__gnu_pbds::sample_trie_access_traits` Struct Reference

```
#include <sample_trie_access_traits.hpp>
```

### Public Types

- enum { `max_size` }
- typedef `std::string::const_iterator` `const_iterator`
- typedef char `e_type`
- typedef `rebind_traits< _Alloc, key_type >::const_reference` `key_const_reference`
- typedef `std::string` `key_type`
- typedef `std::size_t` `size_type`

### Static Public Member Functions

- static `const_iterator` `begin` (`key_const_reference`)
- static `size_type` `e_pos` (`e_type`)
- static `const_iterator` `end` (`key_const_reference`)

### 6.897.1 Detailed Description

A sample trie element access traits.

### 6.897.2 Member Typedef Documentation

#### `e_type`

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
Element type.
```

### 6.897.3 Member Function Documentation

#### `begin()`

```
const_iterator __gnu_pbds::sample_trie_access_traits::begin (
 key_const_reference) [inline], [static]
```

Returns a `const_iterator` to the first element of `r_key`.

**e\_pos()**

```
size_type __gnu_pbds::sample_trie_access_traits::e_pos (
 e_type) [inline], [static]
```

Maps an element to a position.

**end()**

```
const_iterator __gnu_pbds::sample_trie_access_traits::end (
 key_const_reference) [inline], [static]
```

Returns a const\_iterator to the after-last element of r\_key.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

## 6.898 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

```
#include <sample_trie_node_update.hpp>
```

**Public Types**

- typedef std::size\_t **metadata\_type**

**Protected Member Functions**

- [sample\\_trie\\_node\\_update](#) ()
- void [operator](#)() (node\_iterator, node\_const\_iterator) const

**6.898.1 Detailed Description**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

**6.898.2 Constructor & Destructor Documentation****sample\_trie\_node\_update()**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_↵
update () [protected]
```

Default constructor.

**6.898.3 Member Function Documentation****operator>()()**

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
 node_iterator ,
 node_const_iterator) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator node\_it; end\_nd\_it is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)

## 6.899 `__gnu_pbds::sample_update_policy` Struct Reference

```
#include <sample_update_policy.hpp>
```

### Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

### Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

### Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

#### 6.899.1 Detailed Description

A sample list-update policy.

#### 6.899.2 Member Typedef Documentation

##### `metadata_type`

```
typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type [protected]
```

Metadata on which this functor operates.

#### 6.899.3 Constructor & Destructor Documentation

##### `sample_update_policy()` [1/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy ()
```

Default constructor.

Referenced by [sample\\_update\\_policy\(\)](#), and [swap\(\)](#).

##### `sample_update_policy()` [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (
 const sample_update_policy &)
```

Copy constructor.

References [sample\\_update\\_policy\(\)](#).

#### 6.899.4 Member Function Documentation

##### `operator>()()` [1/2]

```
metadata_type __gnu_pbds::sample_update_policy::operator() () const [protected]
```

Creates a metadata object.

**operator()()** [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (
 metadata_reference) const [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

**swap()**

```
void __gnu_pbds::sample_update_policy::swap (
 sample_update_policy & other) [inline]
```

Swaps content.

References [sample\\_update\\_policy\(\)](#).

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

**6.900 \_\_gnu\_parallel::sampling\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for \_\_gnu\_parallel::sampling\_tag:

**Public Member Functions**

- [sampling\\_tag](#) ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

**6.900.1 Detailed Description**

Forces parallel merging with exact splitting, at compile time.

**6.900.2 Member Function Documentation****\_\_get\_num\_threads()**

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

### Returns

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

### set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 __ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 6.901 `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs >` Class Template Reference

```
#include <scoped_allocator>
```

### Public Types

- typedef [\\_\\_traits::const\\_pointer](#) **const\_pointer**
- typedef [\\_\\_traits::const\\_void\\_pointer](#) **const\_void\_pointer**
- typedef [\\_\\_traits::difference\\_type](#) **difference\_type**
- typedef [\\_\\_inner\\_type::\\_\\_type](#) **inner\_allocator\_type**
- typedef [\\_\\_and< typename \\_\\_traits::is\\_always\\_equal, typename allocator\\_traits< \\_InnerAllocs >::is\\_always\\_equal... >::type](#) **is\_always\_equal**
- typedef [\\_OuterAlloc](#) **outer\_allocator\_type**
- typedef [\\_\\_traits::pointer](#) **pointer**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_copy\\_assignment, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_copy\\_assignment... >::type](#) **propagate\_on\_container\_copy\_assignment**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_move\\_assignment, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_move\\_assignment... >::type](#) **propagate\_on\_container\_move\_assignment**
- typedef [\\_\\_or< typename \\_\\_traits::propagate\\_on\\_container\\_swap, typename allocator\\_traits< \\_InnerAllocs >::propagate\\_on\\_container\\_swap... >::type](#) **propagate\_on\_container\_swap**
- typedef [\\_\\_traits::size\\_type](#) **size\_type**
- typedef [\\_\\_traits::value\\_type](#) **value\_type**
- typedef [\\_\\_traits::void\\_pointer](#) **void\_pointer**

### Public Member Functions

- `template<typename _Outer2, typename = _Constructible<_Outer2>>>`  
**scoped\_allocator\_adaptor** ([\\_Outer2](#) &&\_\_outer, const [\\_InnerAllocs](#) &... \_\_inner) noexcept
- **scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) &\_\_other) noexcept
- `template<typename _Outer2, typename = _Constructible<const _Outer2>>>`  
**scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#)< [\\_Outer2](#), [\\_InnerAllocs](#)... > &\_\_other) noexcept

- **scoped\_allocator\_adaptor** (scoped\_allocator\_adaptor &&\_\_other) noexcept
- template<typename \_Outer2, typename = \_Constructible<\_Outer2>>  
**scoped\_allocator\_adaptor** (scoped\_allocator\_adaptor< \_Outer2, \_InnerAllocs... > &&\_\_other) noexcept
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const\_void\_pointer \_\_hint)
- template<typename \_Tp, typename... \_Args>  
void **construct** (\_Tp \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n) noexcept
- template<typename \_Tp>  
void **destroy** (\_Tp \*\_\_p)
- const inner\_allocator\_type & **inner\_allocator** () const noexcept
- inner\_allocator\_type & **inner\_allocator** () noexcept
- size\_type **max\_size** () const
- scoped\_allocator\_adaptor & **operator=** (const scoped\_allocator\_adaptor &)=default
- scoped\_allocator\_adaptor & **operator=** (scoped\_allocator\_adaptor &&)=default
- const outer\_allocator\_type & **outer\_allocator** () const noexcept
- outer\_allocator\_type & **outer\_allocator** () noexcept
- scoped\_allocator\_adaptor **select\_on\_container\_copy\_construction** () const

### Friends

- template<typename...>  
struct **\_\_inner\_type\_impl**
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const scoped\_allocator\_adaptor< \_OutA1, \_InA... > &\_\_a, const scoped\_allocator\_adaptor< \_OutA2, \_InA... > &\_\_b) noexcept

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const scoped\_allocator\_adaptor< \_OutA1, \_InA... > &\_\_a, const scoped\_allocator\_adaptor< \_OutA2, \_InA... > &\_\_b) noexcept

#### 6.901.1 Detailed Description

template<typename \_OuterAlloc, typename... \_InnerAllocs>  
class std::scoped\_allocator\_adaptor< \_OuterAlloc, \_InnerAllocs >

An adaptor to recursively pass an allocator to the objects it constructs.  
The documentation for this class was generated from the following file:

- [scoped\\_allocator](#)

## 6.902 std::seed\_seq Class Reference

```
#include <random>
```

### Public Types

- typedef uint\_least32\_t [result\\_type](#)

## Public Member Functions

- [seed\\_seq](#) () noexcept
- `template<typename _InputIterator>`  
`seed_seq` (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- `seed_seq` (const `seed_seq` &)=delete
- `template<typename _IntType, typename = _Require<is_integral<_IntType>>>`  
`seed_seq` (`std::initializer_list`< \_IntType > \_\_il)
- `template<typename _RandomAccessIterator>`  
void **generate** (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_end)
- `seed_seq` & **operator=** (const `seed_seq` &)=delete
- `template<typename _OutputIterator>`  
void **param** (\_OutputIterator \_\_dest) const
- `size_t` **size** () const noexcept

### 6.902.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Since

C++11

### 6.902.2 Member Typedef Documentation

#### `result_type`

`typedef uint_least32_t std::seed\_seq::result\_type`

The type of the seed vales.

### 6.902.3 Constructor & Destructor Documentation

#### `seed_seq()`

`std::seed_seq::seed_seq () [inline], [noexcept]`

Default constructor.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 6.903 `__gnu_cxx::select1st<_Pair>` Struct Template Reference

```
#include <functional>
```

### Public Types

- `typedef _Pair` [argument\\_type](#)
- `typedef _Pair::first_type` [result\\_type](#)

### Public Member Functions

- `_Pair::first_type` & **operator()** (\_Pair &\_\_x) const
- `template<typename _Pair2>`  
`_Pair2::first_type` & **operator()** (\_Pair2 &\_\_x) const
- `const _Pair::first_type` & **operator()** (const \_Pair &\_\_x) const
- `template<typename _Pair2>`  
`const _Pair2::first_type` & **operator()** (const \_Pair2 &\_\_x) const



### 6.903.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select1st<_Pair>
```

An [SGI extension](#) .

### 6.903.2 Member Typedef Documentation

#### argument\_type

```
typedef _Pair std::unary_function<_Pair, _Pair::first_type>::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Pair::first_type std::unary_function<_Pair, _Pair::first_type>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 6.904 `__gnu_cxx::select2nd<_Pair>` Struct Template Reference

```
#include <functional>
```

### Public Types

- typedef \_Pair [argument\\_type](#)
- typedef \_Pair::second\_type [result\\_type](#)

### Public Member Functions

- \_Pair::second\_type & **operator()** (\_Pair &\_\_x) const
- const \_Pair::second\_type & **operator()** (const \_Pair &\_\_x) const

### 6.904.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select2nd<_Pair>
```

An [SGI extension](#) .

### 6.904.2 Member Typedef Documentation

#### argument\_type

```
typedef _Pair std::unary_function<_Pair, _Pair::second_type>::argument_type [inherited]
argument_type is the type of the argument
```

#### result\_type

```
typedef _Pair::second_type std::unary_function<_Pair, _Pair::second_type>::result_type [inherited]
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 6.905 `__gnu_pbds::detail::select_value_type< Key, Mapped >` Struct Template Reference

```
#include <types_traits.hpp>
```

### Public Types

- typedef `std::pair< const Key, Mapped >` **type**

#### 6.905.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::select_value_type< Key, Mapped >
```

Choose `value_type` to be a key/value pair or just a key.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.906 `__gnu_pbds::detail::select_value_type< Key, null_type >` Struct Template Reference

```
#include <types_traits.hpp>
```

### Public Types

- typedef `Key` **type**

#### 6.906.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::select_value_type< Key, null_type >
```

Specialization for sets where the key is the `value_type`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.907 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference

```
#include <istream>
```

### Public Types

- typedef `__istream_type::__ctype_type` **\_\_ctype\_type**
- typedef `_Traits::int_type` **\_\_int\_type**
- typedef `basic_istream< _CharT, _Traits >` **\_\_istream\_type**
- typedef `basic_streambuf< _CharT, _Traits >` **\_\_streambuf\_type**
- typedef `_Traits traits_type`

### Public Member Functions

- [sentry](#) (`basic_istream< _CharT, _Traits > &__is`, `bool __noskipws=false`)
- [operator bool](#) () const

### 6.907.1 Detailed Description

template<typename \_CharT, typename \_Traits>  
class std::basic\_istream<\_CharT, \_Traits>::sentry

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

### 6.907.2 Member Typedef Documentation

#### traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_istream<_CharT, _Traits>::sentry::traits_type
```

Easy access to dependent types.

### 6.907.3 Constructor & Destructor Documentation

#### sentry()

```
template<typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>::sentry::sentry (
 basic_istream<_CharT, _Traits> & __is,
 bool __noskipws = false) [explicit]
```

The constructor performs all the work.

#### Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__is</code>       | The input stream to guard.            |
| <code>__noskipws</code> | Whether to consume whitespace or not. |

If the stream state is good (`__is.good()` is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if `__noskipws` is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

References [std::basic\\_istream<\\_CharT, \\_Traits>::basic\\_istream\(\)](#), [std::ios\\_base::badbit](#), [std::basic\\_ios<\\_CharT, \\_Traits>::good\(\)](#), and [std::ios\\_base::goodbit](#).

### 6.907.4 Member Function Documentation

#### operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>::sentry::operator bool () const [inline], [explicit]
```

Quick status checking.

## Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`). The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 6.908 std::basic\_ostream< \_CharT, \_Traits >::sentry Class Reference

```
#include <ostream.h>
```

### Public Member Functions

- [sentry](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- [~sentry](#) ()
- [operator bool](#) () const

### 6.908.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >::sentry
```

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

### 6.908.2 Constructor & Destructor Documentation

#### sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
 basic_ostream< _CharT, _Traits > & __os) [explicit]
```

The constructor performs preparatory work.

#### Parameters

|                   |                             |
|-------------------|-----------------------------|
| <code>__os</code> | The output stream to guard. |
|-------------------|-----------------------------|

If the stream state is good (`__os.good()` is true), then if the stream is tied to another output stream, `is-<tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

References [std::basic\\_ostream< \\_CharT, \\_Traits >::basic\\_ostream\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::bad\(\)](#), [std::ios\\_base::failbit](#), [std::basic\\_ios< \\_CharT, \\_Traits >::good\(\)](#), [std::basic\\_ios< \\_CharT, \\_Traits >::setstate\(\)](#), and [std::basic\\_ios< \\_CharT, \\_Traits >::tie\(\)](#).

#### ~sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~~sentry () [inline]
Possibly flushes the stream.
```

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor flushes the output stream.

References [std::ios\\_base::badbit](#), [std::uncaught\\_exception\(\)](#), and [std::ios\\_base::unitbuf](#).

### 6.908.3 Member Function Documentation

#### operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::operator bool () const [inline], [explicit]
Quick status checking.
```

#### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).  
The documentation for this class was generated from the following files:

- [ostream.h](#)
- [ostream.tcc](#)

## 6.909 \_\_gnu\_pbds::sequence\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



### 6.909.1 Detailed Description

Basic sequence.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.910 \_\_gnu\_parallel::sequential\_tag Struct Reference

```
#include <tags.h>
```

### 6.910.1 Detailed Description

Forces sequential execution at compile time.

The documentation for this struct was generated from the following file:

- [tags.h](#)

### 6.911 std::\_\_debug::set< \_Key, \_Compare, \_Allocator > Class Template Reference

```
#include <set.h>
```

Inheritance diagram for std::\_\_debug::set< \_Key, \_Compare, \_Allocator >:



#### Public Types

- typedef \_Allocator **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, set > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, set > **iterator**
- typedef \_Compare **key\_compare**
- typedef \_Key **key\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Key **value\_type**

#### Public Member Functions

- **set** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**set** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- template<typename \_InputIterator>  
**set** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- **set** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- **set** (const allocator\_type &\_\_a)
- **set** (const set &)=default
- **set** (const set &\_\_x, const \_\_type\_identity\_t< allocator\_type > &\_\_a)

- **set** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **set** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a)
- **set** (set &&)=default
- **set** (set &&\_\_x, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(noexcept(\_Base([std::move](#)(\_\_x), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_x) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_position)
- template<\_\_heterogeneous\_tree\_key< set > \_Kt>  
size\_type **erase** (\_Kt &&\_\_x)
- size\_type **erase** (const key\_type &\_\_x)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_position)
- iterator **find** (const key\_type &\_\_x)
- const\_iterator **find** (const key\_type &\_\_x) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<\_\_heterogeneous\_tree\_key< set > \_Kt>  
[std::pair](#)< iterator, bool > **insert** (\_Kt &&\_\_x)
- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_x)
- template<\_\_heterogeneous\_tree\_key< set > \_Kt>  
iterator **insert** (const\_iterator \_\_position, \_Kt &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_x)
- iterator **lower\_bound** (const key\_type &\_\_x)
- const\_iterator **lower\_bound** (const key\_type &\_\_x) const
- set & **operator=** (const set &)=default
- set & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- set & **operator=** (set &&)=default
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- reverse\_iterator **rend** () noexcept
- void **swap** (set &\_\_x) noexcept([/\\*conditional \\*/](#))
- iterator **upper\_bound** (const key\_type &\_\_x)
- const\_iterator **upper\_bound** (const key\_type &\_\_x) const

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::\_Safe\_iterator

#### 6.911.1 Detailed Description

template<typename \_Key, typename \_Compare = std::less<\_Key>, typename \_Allocator = std::allocator<\_Key>>

class std::\_\_debug::set< \_Key, \_Compare, \_Allocator >

Class std::set with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [set.h](#)

## 6.912 std::set< \_Key, \_Compare, \_Alloc > Class Template Reference

```
#include <set>
```

### Public Types

- typedef \_Key [key\\_type](#)
- typedef \_Key [value\\_type](#)
- typedef \_Compare [key\\_compare](#)
- typedef \_Compare [value\\_compare](#)
- typedef \_Alloc [allocator\\_type](#)
- typedef \_Alloc\_traits::pointer [pointer](#)
- typedef \_Alloc\_traits::const\_pointer [const\\_pointer](#)
- typedef \_Alloc\_traits::reference [reference](#)
- typedef \_Alloc\_traits::const\_reference [const\\_reference](#)
- typedef \_Rep\_type::const\_iterator [iterator](#)
- typedef \_Rep\_type::const\_iterator [const\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [reverse\\_iterator](#)
- typedef \_Rep\_type::const\_reverse\_iterator [const\\_reverse\\_iterator](#)
- typedef \_Rep\_type::size\_type [size\\_type](#)
- typedef \_Rep\_type::difference\_type [difference\\_type](#)

### Public Member Functions

- [set](#) ()=default
- template<typename \_InputIterator>  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator>  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator>  
[set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)



- `set` (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=`allocator_type`())
  - `set` (const `allocator_type` &\_\_a)
  - `set` (const `set` &)=default
  - `set` (const `set` &\_\_x, const `__type_identity_t`< `allocator_type` > &\_\_a)
  - `set` (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
  - `set` (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
  - `set` (`set` &&)=default
  - `set` (`set` &&\_\_x, const `__type_identity_t`< `allocator_type` > &\_\_a) noexcept(`is_nothrow_copy_constructible`< `_Compare` >::value && `_Alloc_traits::S_always_equal`())
  - `~set` ()=default
  - `iterator begin` () const noexcept
  - `iterator cbegin` () const noexcept
  - `iterator cend` () const noexcept
  - `void clear` () noexcept
  - `reverse_iterator crbegin` () const noexcept
  - `reverse_iterator crend` () const noexcept
  - `template`<typename... `_Args`>  
`std::pair`< `iterator`, `bool` > `emplace` (`_Args` &&... \_\_args)
  - `template`<typename... `_Args`>  
`iterator emplace_hint` (`const_iterator` \_\_pos, `_Args` &&... \_\_args)
  - `bool empty` () const noexcept
  - `iterator end` () const noexcept
  - `size_type erase` (const `key_type` &\_\_x)
  - `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
  - `iterator erase` (`const_iterator` \_\_position)
  - `allocator_type get_allocator` () const noexcept
  - `template`<typename `_InputIterator`>  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
  - `std::pair`< `iterator`, `bool` > `insert` (const `value_type` &\_\_x)
  - `void insert` (`initializer_list`< `value_type` > \_\_l)
  - `std::pair`< `iterator`, `bool` > `insert` (`value_type` &&\_\_x)
  - `key_compare key_comp` () const
  - `size_type max_size` () const noexcept
  - `set & operator=` (const `set` &)=default
  - `set & operator=` (`initializer_list`< `value_type` > \_\_l)
  - `set & operator=` (`set` &&)=default
  - `reverse_iterator rbegin` () const noexcept
  - `reverse_iterator rend` () const noexcept
  - `size_type size` () const noexcept
  - `void swap` (`set` &\_\_x) noexcept(`/*conditional */`)
  - `value_compare value_comp` () const
- 
- `iterator insert` (`const_iterator` \_\_position, const `value_type` &\_\_x)
  - `iterator insert` (`const_iterator` \_\_position, `value_type` &&\_\_x)
- 
- `size_type count` (const `key_type` &\_\_x) const

- `bool contains (const key_type &__x) const`
- `template<typename _Kt>  
auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`

#### Friends

- `template<typename _K1, typename _C1, typename _A1>  
bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1>  
bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

#### 6.912.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>  
class std::set< _Key, _Compare, _Alloc >`

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Since

C++98

#### Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

### 6.912.2 Member Typedef Documentation

#### allocator\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Alloc std::set<_Key, _Compare, _Alloc>::allocator_type
```

Public typedefs.

#### const\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::const_iterator
```

Iterator-related typedefs.

#### const\_pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Alloc_traits::const_pointer std::set<_Key, _Compare, _Alloc>::const_pointer
```

Iterator-related typedefs.

#### const\_reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Alloc_traits::const_reference std::set<_Key, _Compare, _Alloc>::const_reference
```

Iterator-related typedefs.

#### const\_reverse\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::const_reverse_iterator
```

Iterator-related typedefs.

#### difference\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Rep_type::difference_type std::set<_Key, _Compare, _Alloc>::difference_type
```

Iterator-related typedefs.

#### iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::iterator
```

Iterator-related typedefs.

#### key\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
Public typedefs.
```

### key\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
Public typedefs.
```

### pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
Iterator-related typedefs.
```

### reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
Iterator-related typedefs.
```

### reverse\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
Iterator-related typedefs.
```

### size\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type
Iterator-related typedefs.
```

### value\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::value_compare
Public typedefs.
```

### value\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::value_type
Public typedefs.
```

### 6.912.3 Constructor & Destructor Documentation

#### set() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::set<_Key, _Compare, _Alloc >::set () [default]
```

Default constructor creates no elements.

#### set() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::set<_Key, _Compare, _Alloc >::set (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a set with no elements.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

#### set() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _InputIterator>
```

```
std::set<_Key, _Compare, _Alloc >::set (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a set from a range.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a set consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(__first, __last)`).

#### set() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _InputIterator>
```

```
std::set<_Key, _Compare, _Alloc >::set (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from a range.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

**set()** [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 const set<_Key, _Compare, _Alloc> &) [default]
```

Set copy constructor.

Whether the allocator is copied depends on the allocator traits.

**set()** [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 set<_Key, _Compare, _Alloc> &&) [default]
```

Set move constructor

The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

**set()** [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 initializer_list<value_type> & __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from an `initializer_list`.

**Parameters**

|                     |                                    |
|---------------------|------------------------------------|
| <code>__l</code>    | An <code>initializer_list</code> . |
| <code>__comp</code> | A comparison functor.              |
| <code>__a</code>    | An allocator object.               |

Create a set consisting of copies of the elements in the list. This is linear in  $N$  if the list is already sorted, and  $N\log N$

otherwise (where N is `__l.size()`).

#### set() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

#### set() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 const set<_Key, _Compare, _Alloc> & __x,
 const __type_identity_t< allocator_type > & __a) [inline]
```

Allocator-extended copy constructor.

#### set() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 set<_Key, _Compare, _Alloc> && __x,
 const __type_identity_t< allocator_type > & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

#### set() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

#### set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator>
std::set<_Key, _Compare, _Alloc>::set (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

#### ~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::~set () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### 6.912.4 Member Function Documentation

##### begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

##### cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

##### cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

##### clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
void std::set< _Key, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

##### contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
template<typename _Kt>
```

```
auto std::set< _Key, _Compare, _Alloc >::contains (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

##### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>↵</code>   | Key of elements to be located. |
| <code>__x</code> |                                |

##### Returns

True if there is an element with the specified key.



**contains()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::set<_Key, _Compare, _Alloc>::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is an element with the specified key.

**count()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set<_Key, _Compare, _Alloc>::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

**crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set<_Key, _Compare, _Alloc>::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

**crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set<_Key, _Compare, _Alloc>::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

**emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert an element into the set.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

**emplace\_hint()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to insert an element into the set.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

**empty()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::set< _Key, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

**end()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
iterator std::set<_Key, _Compare, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

**equal\_range()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::pair< iterator, iterator > std::set<_Key, _Compare, _Alloc>::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**equal\_range()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::pair< const_iterator, const_iterator > std::set<_Key, _Compare, _Alloc>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

**erase()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find() [1/2]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a set.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

**find() [2/2]**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a set.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

**insert()** [1/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
template<typename _InputIterator>
```

```
void std::set< _Key, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert an element into the set.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Referenced by `std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert()`.

**insert()** [3/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Attempts to insert an element into the set.

#### Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

If a heterogeneous key `__k` matches a range of elements, an iterator to the first is returned.

Insertion requires logarithmic time (if the hint is not taken).

#### insert() [4/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Attempts to insert an element into the set.

#### Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

#### Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

If a heterogeneous key `__k` matches a range of elements, an iterator to the first is returned.

Insertion requires logarithmic time (if the hint is not taken).

#### insert() [5/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set<_Key, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the set.

**Parameters**

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <code>↵</code>  | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| <code>_↵</code> |                                                                                    |
| <code>↵</code>  |                                                                                    |
| <code>_↵</code> |                                                                                    |
| <code>/</code>  |                                                                                    |

Complexity similar to that of the range constructor.

**key\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
key_compare std::set< _Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

**lower\_bound()** [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**lower\_bound()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |



**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

**max\_size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
size_type std::set<_Key, _Compare, _Alloc>::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

**operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
set & std::set<_Key, _Compare, _Alloc>::operator= (
 const set<_Key, _Compare, _Alloc> &) [default]
```

Set assignment operator.

Whether the allocator is copied depends on the allocator traits.

**operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
set & std::set<_Key, _Compare, _Alloc>::operator= (
 initializer_list<value_type> & __l) [inline]
```

Set list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↵ | An initializer_list. |
| ↵ |                      |
| ↵ |                      |
| ↵ |                      |
| / |                      |

This function fills a set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
set & std::set<_Key, _Compare, _Alloc>::operator= (
 set<_Key, _Compare, _Alloc> &&) [default]
```

Move assignment operator.

**rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

### rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
reverse_iterator std::set< _Key, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

### size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
size_type std::set< _Key, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the set.

### swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
void std::set< _Key, _Compare, _Alloc >::swap (
 set< _Key, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another set.

### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>↵</code>   | A set of the same element and allocator types. |
| <code>__x</code> |                                                |

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Referenced by [std::swap\(\)](#).

### upper\_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
```

```
iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

**Returns**

Iterator pointing to the first element greater than key, or end().

**upper\_bound()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc>::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

**value\_comp()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::set<_Key, _Compare, _Alloc>::value_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

The documentation for this class was generated from the following files:

- [stl\\_multiset.h](#)
- [stl\\_set.h](#)

**6.913 std::shared\_future<\_Res> Class Template Reference**

```
#include <future>
```

Inheritance diagram for std::shared\_future<\_Res>:



## Public Member Functions

- [shared\\_future](#) (const shared\_future &\_\_sf) noexcept
- [shared\\_future](#) (future< \_Res > &&\_\_uf) noexcept
- [shared\\_future](#) (shared\_future &&\_\_sf) noexcept
- const \_Res & [get](#) () const
- shared\_future & **operator=** (const shared\_future &\_\_sf) noexcept
- shared\_future & **operator=** (shared\_future &&\_\_sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename \_Rep, typename \_Period>  
[future\\_status](#) **wait\_for** (const chrono::duration< \_Rep, \_Period > &\_\_rel) const
- template<typename \_Clock, typename \_Duration>  
[future\\_status](#) **wait\_until** (const chrono::time\_point< \_Clock, \_Duration > &\_\_abs) const

## Protected Types

- typedef \_\_future\_base::Result< \_Res > & **\_\_result\_type**
- typedef [shared\\_ptr](#)< \_State\_base > **\_\_state\_type**

## Protected Member Functions

- \_\_result\_type [\\_M\\_get\\_result](#) () const
- void [\\_M\\_swap](#) (\_\_basic\_future &\_\_that) noexcept

### 6.913.1 Detailed Description

```
template<typename _Res>
class std::shared_future< _Res >
```

Primary template for shared\_future.

### 6.913.2 Constructor & Destructor Documentation

#### **shared\_future()** [1/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
 const shared_future< _Res > & __sf) [inline], [noexcept]
```

Copy constructor.

#### **shared\_future()** [2/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
 future< _Res > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

References [std::move\(\)](#).

#### **shared\_future()** [3/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
 shared_future< _Res > && __sf) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

References [std::move\(\)](#).

### 6.913.3 Member Function Documentation

#### M\_get\_result()

```
template<typename _Res>
```

```
__result_type std::__basic_future<_Res>::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by [std::future<\\_Res>::get\(\)](#), and [std::shared\\_future<\\_Res>::get\(\)](#).

#### get()

```
template<typename _Res>
```

```
const _Res & std::shared_future<_Res>::get () const [inline]
```

Retrieving the value.

References [std::\\_\\_basic\\_future<\\_Res>::\\_M\\_get\\_result\(\)](#).

The documentation for this class was generated from the following file:

- [future](#)

## 6.914 std::shared\_future<\_Res &> Class Template Reference

```
#include <future>
```

Inheritance diagram for `std::shared_future<_Res &>`:



### Public Member Functions

- [shared\\_future](#) (const shared\_future &\_\_sf)
- [shared\\_future](#) (future<\_Res &> &&\_\_uf) noexcept
- [shared\\_future](#) (shared\_future &&\_\_sf) noexcept
- \_Res & [get](#) () const
- shared\_future & **operator=** (const shared\_future &\_\_sf)
- shared\_future & **operator=** (shared\_future &&\_\_sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future\_status **wait\_for** (const chrono::duration<\_Rep, \_Period> &\_\_rel) const
- future\_status **wait\_until** (const chrono::time\_point<\_Clock, \_Duration> &\_\_abs) const

### Protected Types

- typedef `__future_base::Result<_Res>` & `__result_type`
- typedef `shared_ptr<_State_base>` `__state_type`

### Protected Member Functions

- `__result_type _M_get_result()` const
- void `_M_swap` (`__basic_future` & `__that`) noexcept

#### 6.914.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res &>
```

Partial specialization for `shared_future<R&>`.

#### 6.914.2 Constructor & Destructor Documentation

##### `shared_future()` [1/3]

```
template<typename _Res>
std::shared_future<_Res &>::shared_future (
 const shared_future<_Res &> & __sf) [inline]
```

Copy constructor.

##### `shared_future()` [2/3]

```
template<typename _Res>
std::shared_future<_Res &>::shared_future (
 future<_Res &> && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

References [std::move\(\)](#).

##### `shared_future()` [3/3]

```
template<typename _Res>
std::shared_future<_Res &>::shared_future (
 shared_future<_Res &> && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

References [std::move\(\)](#).

#### 6.914.3 Member Function Documentation

##### `_M_get_result()`

```
__result_type std::__basic_future<_Res>::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

Referenced by [get\(\)](#).

##### `get()`

```
template<typename _Res>
_Res & std::shared_future<_Res &>::get () const [inline]
```

Retrieving the value.

References [\\_M\\_get\\_result\(\)](#).

The documentation for this class was generated from the following file:

- [future](#)

## 6.915 std::shared\_future< void > Class Reference

```
#include <future>
```

Inheritance diagram for std::shared\_future< void >:



### Public Member Functions

- [shared\\_future](#) (const shared\_future &\_\_sf)
- [shared\\_future](#) (future< void > &&\_\_uf) noexcept
- [shared\\_future](#) (shared\_future &&\_\_sf) noexcept
- void **get** () const
- shared\_future & **operator=** (const shared\_future &\_\_sf)
- shared\_future & **operator=** (shared\_future &&\_\_sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- [future\\_status](#) **wait\_for** (const chrono::duration< \_Rep, \_Period > &\_\_rel) const
- [future\\_status](#) **wait\_until** (const chrono::time\_point< \_Clock, \_Duration > &\_\_abs) const

### Protected Types

- typedef \_\_future\_base::Result< void > & **\_\_result\_type**
- typedef [shared\\_ptr](#)< \_State\_base > **\_\_state\_type**

### Protected Member Functions

- \_\_result\_type **\_M\_get\_result** () const
- void **\_M\_swap** (\_\_basic\_future &\_\_that) noexcept

#### 6.915.1 Detailed Description

Explicit specialization for shared\_future<void>.

### 6.915.2 Constructor & Destructor Documentation

#### **shared\_future()** [1/3]

```
std::shared_future< void >::shared_future (
 const shared_future< void > & __sf) [inline]
```

Copy constructor.

#### **shared\_future()** [2/3]

```
std::shared_future< void >::shared_future (
 future< void > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

References [std::move\(\)](#).

#### **shared\_future()** [3/3]

```
std::shared_future< void >::shared_future (
 shared_future< void > && __sf) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

References [std::move\(\)](#).

### 6.915.3 Member Function Documentation

#### **\_M\_get\_result()**

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

The documentation for this class was generated from the following file:

- [future](#)

### 6.916 std::shared\_lock< \_Mutex > Class Template Reference

```
#include <shared_mutex>
```

#### Public Types

- typedef `_Mutex` **mutex\_type**

#### Public Member Functions

- **shared\_lock** (mutex\_type &\_\_m)
- **shared\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#))
- template<typename \_Rep, typename \_Period>  
  **shared\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rel\_time)
- template<typename \_Clock, typename \_Duration>  
  **shared\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- **shared\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **shared\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **shared\_lock** (shared\_lock &&\_\_sl) noexcept
- **shared\_lock** (shared\_lock const &)=delete
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- shared\_lock & **operator=** (shared\_lock &&\_\_sl) noexcept



- shared\_lock & **operator=** (shared\_lock const &)=delete
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** (shared\_lock &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period>  
bool **try\_lock\_for** (const chrono::duration< \_Rep, \_Period > &\_\_rel\_time)
- template<typename \_Clock, typename \_Duration>  
bool **try\_lock\_until** (const chrono::time\_point< \_Clock, \_Duration > &\_\_abs\_time)
- void **unlock** ()

### 6.916.1 Detailed Description

template<typename \_Mutex>  
class std::shared\_lock< \_Mutex >

shared\_lock

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 6.917 std::shared\_ptr< \_Tp > Class Template Reference

```
#include <memory>
```

### Public Types

- using [element\\_type](#)
- using **weak\_type**

### Public Member Functions

- constexpr [shared\\_ptr](#) () noexcept
- template<typename \_Yp, typename = \_Constructible<\_Yp\*>>  
[shared\\_ptr](#) (\_Yp \*\_\_p)
- template<typename \_Yp, typename \_Deleter, typename = \_Constructible<\_Yp\*, \_Deleter>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d)
- template<typename \_Yp, typename \_Deleter, typename \_Alloc, typename = \_Constructible<\_Yp\*, \_Deleter, \_Alloc>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Yp, typename = \_Constructible<auto\_ptr<\_Yp>>>  
**shared\_ptr** (auto\_ptr< \_Yp > &&\_\_r)
- [shared\\_ptr](#) (const shared\_ptr &) noexcept=default
- template<typename \_Yp, typename = \_Constructible<const shared\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const shared\_ptr< \_Yp > &\_\_r) noexcept
- template<typename \_Yp>  
[shared\\_ptr](#) (const shared\_ptr< \_Yp > &\_\_r, [element\\_type](#) \*\_\_p) noexcept
- template<typename \_Yp, typename = \_Constructible<const weak\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [weak\\_ptr](#)< \_Yp > &\_\_r)
- template<typename \_Deleter>  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d)
- template<typename \_Deleter, typename \_Alloc>  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- constexpr [shared\\_ptr](#) (nullptr\_t) noexcept
- [shared\\_ptr](#) (shared\_ptr &&\_\_r) noexcept

- `template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>`  
`shared_ptr (shared_ptr< _Yp > &&__r) noexcept`
- `template<typename _Yp>`  
`shared_ptr (shared_ptr< _Yp > &&__r, element_type *__p) noexcept`
- `template<typename _Tp1, typename>`  
`shared_ptr (std::auto_ptr< _Tp1 > &&__r)`
- `template<typename _Yp, typename _Del, typename = _Constructible<unique_ptr<_Yp, _Del>>>`  
`shared_ptr (unique_ptr< _Yp, _Del > &&__r)`
- `element_type * get () const noexcept`
- `operator bool () const noexcept`
- `element_type & operator* () const noexcept`
- `element_type * operator-> () const noexcept`
- `template<typename _Yp>`  
`_Assignable< auto_ptr< _Yp > > operator= (auto_ptr< _Yp > &&__r)`
- `shared_ptr & operator= (const shared_ptr &) noexcept=default`
- `template<typename _Yp>`  
`_Assignable< const shared_ptr< _Yp > > operator= (const shared_ptr< _Yp > &&__r) noexcept`
- `shared_ptr & operator= (shared_ptr &&__r) noexcept`
- `template<class _Yp>`  
`_Assignable< shared_ptr< _Yp > > operator= (shared_ptr< _Yp > &&__r) noexcept`
- `template<typename _Yp, typename _Del>`  
`_Assignable< unique_ptr< _Yp, _Del > > operator= (unique_ptr< _Yp, _Del > &&__r)`
- `void reset () noexcept`
- `template<typename _Yp>`  
`_SafeConv< _Yp > reset (_Yp *__p)`
- `template<typename _Yp, typename _Deleter>`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d)`
- `template<typename _Yp, typename _Deleter, typename _Alloc>`  
`_SafeConv< _Yp > reset (_Yp *__p, _Deleter __d, _Alloc __a)`
- `void swap (__shared_ptr< _Tp, _Lp > &__other) noexcept`
- `bool unique () const noexcept`
- `long use_count () const noexcept`
  
- `template<typename _Tp1>`  
`bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const noexcept`
- `template<typename _Tp1>`  
`bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const noexcept`

## Friends

- `template<typename _Yp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Yp > > allocate_shared (const _Alloc &, _Args &&...)`
- `template<typename _Yp, typename... _Args>`  
`shared_ptr< _NonArray< _Yp > > make_shared (_Args &&...)`
- `class weak_ptr< _Tp >`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Del, typename _Tp>`  
`_Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept`

- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`  
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Up>`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp>`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp>`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Up>`  
`shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)`

### 6.917.1 Detailed Description

**template<typename \_Tp>**  
**class std::shared\_ptr< \_Tp >**

A smart pointer with reference-counted copy semantics.

Since

C++11

A `shared_ptr` object is either empty or *owns* a pointer passed to the constructor. Copies of a `shared_ptr` share ownership of the same pointer. When the last `shared_ptr` that owns the pointer is destroyed or reset, the owned pointer is freed (either by `delete` or by invoking a custom deleter that was passed to the constructor).

A `shared_ptr` also stores another pointer, which is usually (but not always) the same pointer as it owns. The stored pointer can be retrieved by calling the `get()` member function.

The equality and relational operators for `shared_ptr` only compare the stored pointer returned by `get()`, not the owned pointer. To test whether two `shared_ptr` objects share ownership of the same pointer see `std::shared_ptr::owner_before` and `std::owner_less`.

## 6.917.2 Member Typedef Documentation

### element\_type

```
template<typename _Tp>
```

```
using std::shared_ptr< _Tp >::element_type
```

The type pointed to by the stored pointer, `remove_extent_t<_Tp>`.

## 6.917.3 Constructor & Destructor Documentation

### shared\_ptr() [1/14]

```
template<typename _Tp>
```

```
std::shared_ptr< _Tp >::shared_ptr () [inline], [constexpr], [noexcept]
```

Construct an empty `shared_ptr`.

#### Postcondition

```
use_count()==0 && get()==0
```

### shared\_ptr() [2/14]

```
template<typename _Tp>
```

```
std::shared_ptr< _Tp >::shared_ptr (
 const shared_ptr< _Tp > &) [default], [noexcept]
```

Copy constructor.

### shared\_ptr() [3/14]

```
template<typename _Tp>
```

```
template<typename _Yp, typename = _Constructible<_Yp*>>
```

```
std::shared_ptr< _Tp >::shared_ptr (
 _Yp * __p) [inline], [explicit]
```

Construct a `shared_ptr` that owns the pointer `__p`.

#### Parameters

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__p</code> | A pointer that is convertible to <code>element_type*</code> . |
|------------------|---------------------------------------------------------------|

#### Postcondition

```
use_count() == 1 && get() == __p
```

#### Exceptions

|                                                  |                                               |
|--------------------------------------------------|-----------------------------------------------|
| <code><a href="#">std::bad_alloc</a></code> , in | which case <code>delete __p</code> is called. |
|--------------------------------------------------|-----------------------------------------------|

### shared\_ptr() [4/14]

```
template<typename _Tp>
```

```
template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
```

```
std::shared_ptr< _Tp >::shared_ptr (
```

```

 __Yp * __p,
 _Deleter __d) [inline]

```

Construct a shared\_ptr that owns the pointer `__p` and the deleter `__d`.

#### Parameters

|                  |            |
|------------------|------------|
| <code>__p</code> | A pointer. |
| <code>__d</code> | A deleter. |

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw  
`__shared_ptr` will release `__p` by calling `__d(__p)`

#### shared\_ptr() [5/14]

```

template<typename _Tp>
template<typename _Deleter>
std::shared_ptr<_Tp>::shared_ptr (
 nullptr_t __p,
 _Deleter __d) [inline]

```

Construct a shared\_ptr that owns a null pointer and the deleter `__d`.

#### Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__p</code> | A null pointer constant. |
| <code>__d</code> | A deleter.               |

#### Postcondition

`use_count() == 1 && get() == __p`

#### Exceptions

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw  
The last owner will call `__d(__p)`

**shared\_ptr()** [6/14]

```

template<typename _Tp>
template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*, _↵
Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
 _Yp * __p,
 _Deleter __d,
 _Alloc __a) [inline]

```

Construct a shared\_ptr that owns the pointer `__p` and the deleter `__d`.

**Parameters**

|                        |               |
|------------------------|---------------|
| <code>↵<br/>__p</code> | A pointer.    |
| <code>↵<br/>__d</code> | A deleter.    |
| <code>↵<br/>__a</code> | An allocator. |

**Postcondition**

`use_count() == 1 && get() == __p`

**Exceptions**

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

`__shared_ptr` will release `__p` by calling `__d(__p)`

**shared\_ptr()** [7/14]

```

template<typename _Tp>
template<typename _Deleter, typename _Alloc>
std::shared_ptr< _Tp >::shared_ptr (
 nullptr_t __p,
 _Deleter __d,
 _Alloc __a) [inline]

```

Construct a shared\_ptr that owns a null pointer and the deleter `__d`.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>↵<br/>__p</code> | A null pointer constant. |
| <code>↵<br/>__d</code> | A deleter.               |
| <code>↵<br/>__a</code> | An allocator.            |

**Postcondition**

```
use_count() == 1 && get() == __p
```

**Exceptions**

|                                     |                                             |
|-------------------------------------|---------------------------------------------|
| <a href="#">std::bad_alloc</a> , in | which case <code>__d(__p)</code> is called. |
|-------------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

**shared\_ptr()** [8/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr<_Tp>::shared_ptr (
 const shared_ptr<_Yp> & __r,
 element_type * __p) [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

**Parameters**

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__r</code> | A <code>shared_ptr</code> .                                        |
| <code>__p</code> | A pointer that will remain valid while <code>*__r</code> is valid. |

**Postcondition**

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

**shared\_ptr()** [9/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr<_Tp>::shared_ptr (
 shared_ptr<_Yp> && __r,
 element_type * __p) [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

**Parameters**

|                       |                                                                    |
|-----------------------|--------------------------------------------------------------------|
| $\leftarrow$<br>$\_r$ | A <code>shared_ptr</code> .                                        |
| $\leftarrow$<br>$\_p$ | A pointer that will remain valid while <code>*__r</code> is valid. |

**Postcondition**

```
get() == __p && !__r.use_count() && !__r.get()
```

**Since**

C++17

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi1(pii, &pii->first);
assert(pii.use_count() == 2);
shared_ptr<int> pi2(std::move(pii), &pii->second);
assert(pii.use_count() == 0);
```

**shared\_ptr()** [10/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr<_Tp>::shared_ptr (
 const shared_ptr<_Yp> & __r) [inline], [noexcept]
```

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

**Parameters**

|                       |                             |
|-----------------------|-----------------------------|
| $\leftarrow$<br>$\_r$ | A <code>shared_ptr</code> . |
|-----------------------|-----------------------------|

**Postcondition**

```
get() == __r.get() && use_count() == __r.use_count()
```

**shared\_ptr()** [11/14]

```
template<typename _Tp>
std::shared_ptr<_Tp>::shared_ptr (
 shared_ptr<_Tp> && __r) [inline], [noexcept]
```

Move-constructs a `shared_ptr` instance from `__r`.



**Parameters**

|          |                      |
|----------|----------------------|
| ↩        | A shared_ptr rvalue. |
| ↩        |                      |
| ↩        |                      |
| ↩        |                      |
| <i>r</i> |                      |

**Postcondition**

\*this contains the old value of \_\_r, \_\_r is empty.

**shared\_ptr() [12/14]**

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr<_Tp>::shared_ptr (
 shared_ptr<_Yp> && __r) [inline], [noexcept]
```

Move-constructs a shared\_ptr instance from \_\_r.

**Parameters**

|          |                      |
|----------|----------------------|
| ↩        | A shared_ptr rvalue. |
| ↩        |                      |
| ↩        |                      |
| ↩        |                      |
| <i>r</i> |                      |

**Postcondition**

\*this contains the old value of \_\_r, \_\_r is empty.

**shared\_ptr() [13/14]**

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
std::shared_ptr<_Tp>::shared_ptr (
 const weak_ptr<_Yp> & __r) [inline], [explicit]
```

Constructs a shared\_ptr that shares ownership with \_\_r and stores a copy of the pointer stored in \_\_r.

**Parameters**

|          |             |
|----------|-------------|
| ↩        | A weak_ptr. |
| ↩        |             |
| ↩        |             |
| ↩        |             |
| <i>r</i> |             |

**Postcondition**

`use_count() == __r.use_count()`

**Exceptions**

|                           |                                                                                |
|---------------------------|--------------------------------------------------------------------------------|
| <code>bad_weak_ptr</code> | when <code>__r.expired()</code> , in which case the constructor has no effect. |
|---------------------------|--------------------------------------------------------------------------------|

**shared\_ptr()** [14/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
 nullptr_t) [inline], [constexpr], [noexcept]
```

Construct an empty `shared_ptr`.

**Postcondition**

`use_count() == 0 && get() == nullptr`

**6.917.4 Member Function Documentation****get()**

```
template<typename _Tp, _Lock_policy _Lp>
element_type * std::__shared_ptr< _Tp, _Lp >::get () const [inline], [noexcept], [inherited]
```

Return the stored pointer.

**operator bool()**

```
template<typename _Tp, _Lock_policy _Lp>
std::__shared_ptr< _Tp, _Lp >::operator bool () const [inline], [explicit], [noexcept], [inherited]
```

Return true if the stored pointer is not null.

**owner\_before()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
 __shared_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

**owner\_before()** [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
 __weak_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

**swap()**

```
template<typename _Tp, _Lock_policy _Lp>
void std::__shared_ptr< _Tp, _Lp >::swap (
 __shared_ptr< _Tp, _Lp > & __other) [inline], [noexcept], [inherited]
```

Exchange both the owned pointer and the stored pointer.

**unique()**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::__shared_ptr< _Tp, _Lp >::unique () const [inline], [noexcept], [inherited]
```

Return true if use\_count() == 1.

**use\_count()**

```
template<typename _Tp, _Lock_policy _Lp>
long std::__shared_ptr< _Tp, _Lp >::use_count () const [inline], [noexcept], [inherited]
```

If \*this owns a pointer, return the number of owners, otherwise zero.

The documentation for this class was generated from the following files:

- [bits/shared\\_ptr.h](#)
- [auto\\_ptr.h](#)

**6.918 std::shared\_timed\_mutex Class Reference**

```
#include <shared_mutex>
```

**Public Member Functions**

- **shared\_timed\_mutex** (const shared\_timed\_mutex &)=delete
- void **lock** ()
- void **lock\_shared** ()
- shared\_timed\_mutex & **operator=** (const shared\_timed\_mutex &)=delete
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- bool **try\_lock\_shared** ()
- template<typename \_Rep, typename \_Period>  
bool **try\_lock\_shared\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
bool **try\_lock\_shared\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- template<typename \_Clock, typename \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- void **unlock** ()
- void **unlock\_shared** ()

**6.918.1 Detailed Description**

The standard shared timed mutex type.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 6.919 `std::shuffle_order_engine<_RandomNumberEngine, __k>` Class Template Reference

```
#include <random>
```

### Public Types

- `template<typename _Sseq>`  
using `_If_seed_seq`
- `typedef _RandomNumberEngine::result_type` `result_type`

### Public Member Functions

- `shuffle_order_engine()`
- `shuffle_order_engine(_RandomNumberEngine &&__rng)`
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`shuffle_order_engine(_Sseq &__q)`
- `shuffle_order_engine(const _RandomNumberEngine &__rng)`
- `shuffle_order_engine(result_type __s)`
- `const _RandomNumberEngine &base()` `const` `noexcept`
- `void discard(unsigned long long __z)`
- `result_type operator()()`
- `void seed()`
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq> seed(_Sseq &__q)`
- `void seed(result_type __s)`

### Static Public Member Functions

- `static constexpr result_type max()`
- `static constexpr result_type min()`

### Static Public Attributes

- `static constexpr size_t table_size`

### Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`std::basic_ostream<_CharT, _Traits> &operator<< (std::basic_ostream<_CharT, _Traits> &__os, const std::shuffle_order_engine<_RandomNumberEngine1, __k1> &__x)`
- `bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)`
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`std::basic_istream<_CharT, _Traits> &operator>> (std::basic_istream<_CharT, _Traits> &__is, std::shuffle_order_engine<_RandomNumberEngine1, __k1> &__x)`

#### 6.919.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine<_RandomNumberEngine, __k>
```

Produces random numbers by reordering random numbers from some base engine.

The values from the base engine are stored in a sequence of size `__k` and shuffled by an algorithm that depends on those values.

Since

C++11

## 6.919.2 Member Typedef Documentation

### result\_type

```
template<typename _RandomNumberEngine, size_t __k>
typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type
```

The type of the generated random value.

## 6.919.3 Constructor & Destructor Documentation

### shuffle\_order\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine () [inline]
```

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

### shuffle\_order\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

### shuffle\_order\_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

### shuffle\_order\_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 result_type __s) [inline], [explicit]
```

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with `__s`.

**Parameters**

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | A seed value for the base class engine. |
| <code>_s</code> |                                         |

**shuffle\_order\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 _Sseq & __q) [inline], [explicit]
```

Generator construct a shuffle\_order\_engine engine.

**Parameters**

|                 |                  |
|-----------------|------------------|
| <code>_↵</code> | A seed sequence. |
| <code>_q</code> |                  |

**6.919.4 Member Function Documentation****base()**

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine & std::shuffle_order_engine< _RandomNumberEngine, __k >::base () const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

**discard()**

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _RandomNumberEngine, size_t __k>
constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max () [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

**min()**

```
template<typename _RandomNumberEngine, size_t __k>
constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min () [inline],
[static], [constexpr]
```

Gets the minimum value in the generated random number range.

**operator>()**

```
template<typename _RandomNumberEngine, size_t __k>
shuffle_order_engine< _RandomNumberEngine, __k >::result_type std::shuffle_order_engine< _↵
RandomNumberEngine, __k >::operator() ()
```

Gets the next value in the generated random number sequence.

References [std::max\(\)](#), and [std::min\(\)](#).

#### seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed () [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

#### seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
 _Sseq & __q) [inline]
```

Reseeds the shuffle\_order\_engine object with the given seed sequence.

#### Parameters

|                                  |                            |
|----------------------------------|----------------------------|
| <a href="#"><code>__q</code></a> | A seed generator function. |
|----------------------------------|----------------------------|

#### seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
 result_type __s) [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

### 6.919.5 Friends And Related Symbol Documentation

#### operator<<

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]
```

Inserts the current state of a shuffle\_order\_engine random number generator engine `__x` into the output stream `__os`.

#### Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
 const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
 const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs) [friend]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

**Parameters**

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A shuffle_order_engine random number generator object.       |
| <code>__rhs</code> | Another shuffle_order_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

**Parameters**

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__is</code> | An input stream.                                       |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

**Returns**

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.920 std::ranges::single\_view< \_Tp > Class Template Reference**

```
#include <ranges>
```



Inheritance diagram for std::ranges::single\_view< \_Tp >:



### Public Member Functions

- constexpr **single\_view** (\_Tp &&\_\_t) noexcept(is\_nothrow\_move\_constructible\_v< \_Tp >)
- constexpr **single\_view** (const \_Tp &\_\_t) noexcept(is\_nothrow\_copy\_constructible\_v< \_Tp >)
- template<typename... \_Args>  
constexpr **single\_view** (in\_place\_t, \_Args &&... \_\_args) noexcept(is\_nothrow\_constructible\_v< \_Tp, \_Args... >)
- constexpr decltype(auto) **back** ()
- constexpr const \_Tp \* **begin** () const noexcept
- constexpr \_Tp \* **begin** () noexcept
- constexpr auto **cbegin** ()
- constexpr auto **cend** ()
- constexpr const \_Tp \* **data** () const noexcept
- constexpr \_Tp \* **data** () noexcept
- constexpr const \_Tp \* **end** () const noexcept
- constexpr \_Tp \* **end** () noexcept
- constexpr decltype(auto) **front** ()
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(\_M\_derived())))
- constexpr decltype(auto) **operator[]** (range\_difference\_t< \_Range > \_\_n)

### Static Public Member Functions

- static constexpr bool **empty** () noexcept
- static constexpr size\_t **size** () noexcept

#### 6.920.1 Detailed Description

```

template<copy_constructible _Tp>
requires is_object_v<_Tp>
class std::ranges::single_view< _Tp >

```

A view that contains exactly one element.

The documentation for this class was generated from the following file:

- [ranges](#)

## 6.921 std::slice Class Reference

```
#include <slice_array.h>
```

### Public Member Functions

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

### Friends

- bool [operator==](#) (const [slice](#) &, const [slice](#) &)=default

#### 6.921.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

#### 6.921.2 Friends And Related Symbol Documentation

##### [operator==](#)

```
bool operator== (
 const slice & ,
 const slice &) [friend]
```

Equality comparison.

References [slice\(\)](#).

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

## 6.922 std::slice\_array< \_Tp > Class Template Reference

```
#include <slice_array.h>
```

### Public Types

- typedef \_Tp **value\_type**

### Public Member Functions

- [slice\\_array](#) (const slice\_array &)
- template<class \_Dom>  
void **operator%=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator%=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom>  
void **operator&=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator&=** (const [valarray](#)< \_Tp > &) const

- template<class \_Dom>  
void **operator**\*= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**\*= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**+= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**+= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**-= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**-= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**/= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**/= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**<<= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**<<= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**= (const \_Tp &) const
- slice\_array & **operator**= (const slice\_array &)
- void **operator**= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**>>= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**>>= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**^= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**^= (const valarray< \_Tp > &) const
- template<class \_Dom>  
void **operator**|= (const \_Expr< \_Dom, \_Tp > &) const
- void **operator**|= (const valarray< \_Tp > &) const

## Friends

- class valarray< \_Tp >

### 6.922.1 Detailed Description

template<typename \_Tp>  
class std::slice\_array< \_Tp >

Reference to one-dimensional subset of an array.

A slice\_array is a reference to the actual elements of an array specified by a slice. The way to get a slice\_array is to call operator[](slice) on a valarray. The returned slice\_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice\_array refers to.

## Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

### 6.922.2 Member Function Documentation

#### **operator%=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator%= (
 const valarray< _Tp > &) const
```

Modulo slice elements by corresponding elements of *v*.

#### **operator&=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator&= (
 const valarray< _Tp > &) const
```

Logical and slice elements with corresponding elements of *v*.

#### **operator\*=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator*= (
 const valarray< _Tp > &) const
```

Multiply slice elements by corresponding elements of *v*.

#### **operator+=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator+= (
 const valarray< _Tp > &) const
```

Add corresponding elements of *v* to slice elements.

#### **operator-=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator-= (
 const valarray< _Tp > &) const
```

Subtract corresponding elements of *v* from slice elements.

#### **operator/=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator/= (
 const valarray< _Tp > &) const
```

Divide slice elements by corresponding elements of *v*.

#### **operator<<=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator<<= (
 const valarray< _Tp > &) const
```

Left shift slice elements by corresponding elements of *v*.

#### **operator>>=( )**

```
template<typename _Tp>
void std::slice_array< _Tp >::operator>>= (
 const valarray< _Tp > &) const
```

Right shift slice elements by corresponding elements of *v*.

### `operator^=()`

```
template<typename _Tp>
void std::slice_array<_Tp>::operator^= (
 const valarray<_Tp> &) const
```

Logical xor slice elements with corresponding elements of *v*.

### `operator" |=()`

```
template<typename _Tp>
void std::slice_array<_Tp>::operator|= (
 const valarray<_Tp> &) const
```

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following files:

- [valarray](#)
- [slice\\_array.h](#)

## 6.923 `__gnu_cxx::slist<_Tp, _Alloc>` Class Template Reference

```
#include <slist>
```

### Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `_Slist_iterator<_Tp, const _Tp &, const _Tp * >` **const\_iterator**
- typedef `const value_type *` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef `_Slist_iterator<_Tp, _Tp &, _Tp * >` **iterator**
- typedef `value_type *` **pointer**
- typedef `value_type &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

### Public Member Functions

- template<class `_InputIterator`>  
    **slist** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a=allocator\_type())
- **slist** (const `allocator_type` &\_\_a=allocator\_type())
- **slist** (const `slist` &\_\_x)
- **slist** (size\_type \_\_n)
- **slist** (size\_type \_\_n, const `value_type` &\_\_x, const `allocator_type` &\_\_a=allocator\_type())
- template<class `_InputIterator`>  
    void **\_M\_assign\_dispatch** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, std::\_\_false\_type)
- template<class `_Integer`>  
    void **\_M\_assign\_dispatch** (`_Integer` \_\_n, `_Integer` \_\_val, std::\_\_true\_type)
- void **\_M\_fill\_assign** (size\_type \_\_n, const `_Tp` &\_\_val)
- template<class `_InputIterator`>  
    void **assign** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void **assign** (size\_type \_\_n, const `_Tp` &\_\_val)
- iterator **before\_begin** ()

- const\_iterator **before\_begin** () const
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator \_\_first, iterator \_\_last)
- iterator **erase** (iterator \_\_pos)
- iterator **erase\_after** (iterator \_\_before\_first, iterator \_\_last)
- iterator **erase\_after** (iterator \_\_pos)
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (iterator \_\_pos)
- template<class \_InIterator>  
void **insert** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert** (iterator \_\_pos, const value\_type &\_\_x)
- void **insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- iterator **insert\_after** (iterator \_\_pos)
- template<class \_InIterator>  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- iterator **insert\_after** (iterator \_\_pos, const value\_type &\_\_x)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- size\_type **max\_size** () const
- template<class \_StrictWeakOrdering>  
void **merge** (slist &, \_StrictWeakOrdering)
- void **merge** (slist &\_\_x)
- slist & **operator=** (const slist &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** ()
- void **push\_front** (const value\_type &\_\_x)
- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate>  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering>  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, slist &\_\_x)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, slist &\_\_x)
- void **swap** (slist &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate>  
void **unique** (\_BinaryPredicate \_\_pred)

### Static Private Member Functions

- static constexpr size\_type [max\\_size](#) (const \_Alloc &\_\_a) noexcept

#### 6.923.1 Detailed Description

```
template<class _Tp, class _Alloc = std::allocator<_Tp>>
class __gnu_cxx::slist< _Tp, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/↵documentation_style.html)

The documentation for this class was generated from the following file:

- [slist](#)

## 6.924 std::experimental::filesystem::v1::space\_info Struct Reference

```
#include <fs_fwd.h>
```

### Public Attributes

- uintmax\_t **available**
- uintmax\_t **capacity**
- uintmax\_t **free**

#### 6.924.1 Detailed Description

Information about free space on a disk.

The documentation for this struct was generated from the following file:

- [experimental/bits/fs\\_fwd.h](#)

## 6.925 std::filesystem::space\_info Struct Reference

```
#include <fs_fwd.h>
```

### Public Attributes

- uintmax\_t **available**
- uintmax\_t **capacity**
- uintmax\_t **free**

### Friends

- bool **operator==** (const [space\\_info](#) &, const [space\\_info](#) &)=default

#### 6.925.1 Detailed Description

Information about free space on a disk.

The documentation for this struct was generated from the following file:

- [bits/fs\\_fwd.h](#)

## 6.926 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

```
#include <splay_tree_.hpp>
```

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `splay_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `base_type::node_update` **node\_update**
- typedef `base_type::const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `base_type::value_type` **value\_type**

### Public Member Functions

- **splay\_tree\_map** (const `Cmp_Fn` &)
- **splay\_tree\_map** (const `Cmp_Fn` &, const `node_update` &)
- **splay\_tree\_map** (const `splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename `It`>  
void **copy\_from\_range** (`It`, `It`)
- bool **empty** () const



- iterator **end** ()
- const\_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key\_const\_reference)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **initialize** ()
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** (splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_iterator [node\\_begin](#) ()
- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_end](#) ()
- node\_const\_iterator [node\\_end](#) () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_alloc\_traits::value\_type **node**
- typedef node\_alloc\_traits::allocator\_type **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_subtree\_size** (node\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

#### 6.926.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Splay tree.

## 6.926.2 Member Function Documentation

### node\_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin() [inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

### node\_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin() const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to the node at the root of the tree.

### node\_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end() [inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

### node\_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

```
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end() const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- [splay\\_tree.hpp](#)

## 6.927 \_\_gnu\_pbds::detail::splay\_tree\_node\_< Value\_Type, Metadata, \_Alloc > Struct Template Reference

```
#include <node.hpp>
```

### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::reference **metadata\_reference**
- typedef Metadata **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, [splay\\_tree\\_node\\_](#) >::pointer **node\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, [splay\\_tree\\_node\\_](#) >::size\_type **size\_type**
- typedef Value\_Type **value\_type**

### Public Member Functions

- metadata\_reference **get\_metadata** ()
- metadata\_const\_reference **get\_metadata** () const
- bool **special** () const

### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_special**
- size\_type **m\_subtree\_size**
- value\_type **m\_value**

#### 6.927.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for splay tree.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/node.hpp](#)

#### 6.928 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::splay_tree_tag`:



### 6.928.1 Detailed Description

Splay tree.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.929 `std::stack<_Tp, _Sequence>` Class Template Reference

```
#include <stl_stack.h>
```

### Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

### Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>`  
[stack](#) ()

- `template<typename _InputIterator, typename _Alloc, typename = _RequireInputIter<_InputIterator>, typename = _Uses<_Alloc>>`  
**stack** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Alloc &\_\_a)
- **stack** (\_Sequence &&\_\_c)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (\_Sequence &&\_\_c, const \_Alloc &\_\_a)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (const \_Alloc &\_\_a)
- **stack** (const \_Sequence &&\_\_c)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (const \_Sequence &&\_\_c, const \_Alloc &\_\_a)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (const stack &&\_\_q, const \_Alloc &\_\_a)
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
**stack** (stack &&\_\_q, const \_Alloc &\_\_a)
- `template<typename... _Args>`  
`decltype(auto) emplace (_Args &&... __args)`
- `bool empty () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (stack &&__s) noexcept(__is_nothrow_swappable<_Sequence>::value)`
- `reference top ()`
- `const_reference top () const`

## Protected Attributes

- `_Sequence c`

## Friends

- `template<typename _Tp1, typename _Seq1>`  
`bool operator< (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`
- `template<typename _Tp1, three_way_comparable _Seq1>`  
`compare_three_way_result_t<_Seq1> operator<=> (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`
- `template<typename _Tp1, typename _Seq1>`  
`bool operator== (const stack<_Tp1, _Seq1> &, const stack<_Tp1, _Seq1> &)`

### 6.929.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>>`  
**class** `std::stack<_Tp, _Sequence>`

A standard container giving FILO behavior.

## Template Parameters

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                         |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

### 6.929.2 Constructor & Destructor Documentation

#### stack()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::stack<_Tp, _Sequence>::stack () [inline]
```

Default constructor creates no elements.

### 6.929.3 Member Function Documentation

#### empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack<_Tp, _Sequence>::empty () const [inline], [nodiscard]
```

Returns true if the stack is empty.

#### pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence>::pop () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

#### push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence>::push (
 const value_type & __x) [inline]
```

Add data to the top of the stack.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

#### size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack<_Tp, _Sequence>::size () const [inline], [nodiscard]
```

Returns the number of elements in the stack.

**top()** [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence>::top () [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the stack.

**top()** [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence>::top () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the stack.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

**6.930 \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits> Class Template Reference**

```
#include <stdio_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:

**Public Types**

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic\_filebuf< char_type, traits_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic\_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`



- typedef std::size\_t **size\_t**
- typedef \_Traits **traits\_type**

### Public Member Functions

- `stdio_filebuf` ()
- `stdio_filebuf` (int \_\_fd, std::ios\_base::openmode \_\_mode, size\_t \_\_size=static\_cast< size\_t >(BUFSIZ))
- `stdio_filebuf` (std::\_\_c\_file \* \_\_f, std::ios\_base::openmode \_\_mode, size\_t \_\_size=static\_cast< size\_t >(BUFSIZ))
- `stdio_filebuf` (stdio\_filebuf &&)=default
- virtual `~stdio_filebuf` ()
- `__filebuf_type` \* `close` ()
- int `fd` ()
- std::\_\_c\_file \* `file` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- bool `is_open` () const throw ()
- template<typename \_Path>  
\_If\_fs\_path< \_Path, \_\_filebuf\_type \* > `open` (const \_Path & \_\_s, ios\_base::openmode \_\_mode)
- `__filebuf_type` \* `open` (const char \* \_\_s, ios\_base::openmode \_\_mode)
- `__filebuf_type` \* `open` (const std::string & \_\_s, ios\_base::openmode \_\_mode)
- `stdio_filebuf` & `operator=` (stdio\_filebuf &&)=default
- locale `pubimbue` (const locale & \_\_loc)
- int\_type `sputc` ()
- int\_type `sgetc` ()
- streamsize `sgetn` (char\_type \* \_\_s, streamsize \_\_n)
- int\_type `snextc` ()
- int\_type `sputbackc` (char\_type \_\_c)
- int\_type `sputc` (char\_type \_\_c)
- streamsize `sputn` (const char\_type \* \_\_s, streamsize \_\_n)
- int\_type `sungetc` ()
- void `swap` (stdio\_filebuf & \_\_fb)
- void `swap` (basic\_filebuf &)
- `basic_streambuf` \* `pubsetbuf` (char\_type \* \_\_s, streamsize \_\_n)
- pos\_type `pubseekoff` (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- pos\_type `pubseekpos` (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- int `pubsync` ()

### Protected Member Functions

- void `__safe_gbump` (streamsize \_\_n)
- void `__safe_pbump` (streamsize \_\_n)
- void `_M_allocate_internal_buffer` ()
- bool `_M_convert_to_external` (char\_type \*, streamsize)
- void `_M_create_pback` ()
- void `_M_destroy_internal_buffer` () throw ()
- void `_M_destroy_pback` () throw ()
- int `_M_get_ext_pos` (\_\_state\_type & \_\_state)
- pos\_type `_M_seek` (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
- void `_M_set_buffer` (streamsize \_\_off)

- `bool _M_terminate_output ()`
  - `void gbump (int __n)`
  - `virtual void imbue (const locale & __loc)`
  - `virtual int_type overflow (int_type __c= Traits::eof())`
  - `virtual int_type pbackfail (int_type __c= Traits::eof())`
  - `void pbump (int __n)`
  - `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual __streambuf_type * setbuf (char_type * __s, streamsize __n)`
  - `void setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)`
  - `void setp (char_type * __pbeg, char_type * __pend)`
  - `virtual streamsize showmanyc ()`
  - `void swap (basic_streambuf & __sb)`
  - `virtual int sync ()`
  - `virtual int_type uflow ()`
  - `virtual int_type underflow ()`
  - `virtual streamsize xsgetn (char_type * __s, streamsize __n)`
  - `virtual streamsize xsputn (const char_type * __s, streamsize __n)`
- 
- `char_type * eback () const`
  - `char_type * gptr () const`
  - `char_type * egptr () const`
- 
- `char_type * pbase () const`
  - `char_type * pptr () const`
  - `char_type * epptr () const`

### Protected Attributes

- `char_type * _M_buf`
- `bool _M_buf_allocated`
- `locale _M_buf_locale`
- `size_t _M_buf_size`
- `const __codecvt_type * _M_codecvt`
- `char * _M_ext_buf`
- `streamsize _M_ext_buf_size`
- `char * _M_ext_end`
- `const char * _M_ext_next`
- `__file_type _M_file`
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `__c_lock _M_lock`
- `ios_base::openmode _M_mode`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `bool _M_reading`

- `__state_type _M_state_beg`
- `__state_type _M_state_cur`
- `__state_type _M_state_last`
- `bool _M_writing`
- `char_type _M_pback`
- `char_type * _M_pback_cur_save`
- `char_type * _M_pback_end_save`
- `bool _M_pback_init`

### 6.930.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

### 6.930.2 Constructor & Destructor Documentation

#### `stdio_filebuf()` [1/3]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf () [inline]
deferred initialization
References std::basic_filebuf<_CharT, _Traits>::basic_filebuf\(\).
```

#### `stdio_filebuf()` [2/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
 int __fd,
 std::ios_base::openmode __mode,
 size_t __size = static_cast<size_t>(BUFSIZ))
```

#### Parameters

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__fd</code>   | An open file descriptor.                                |
| <code>__mode</code> | Same meaning as in a standard filebuf.                  |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. |

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

References [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_buf\\_size](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_mode](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_reading](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_set\\_buffer\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::is\\_open\(\)](#).

#### `stdio_filebuf()` [3/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
 std::__c_file * __f,
```

```
std::ios_base::openmode __mode,
size_t __size = static_cast<size_t>(BUFSIZ))
```

### Parameters

|                     |                                                                                      |
|---------------------|--------------------------------------------------------------------------------------|
| <code>__f</code>    | An open FILE*.                                                                       |
| <code>__mode</code> | Same meaning as in a standard filebuf.                                               |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. Defaults to system's BUFSIZ. |

This constructor associates a file stream buffer with an open C FILE\*. The FILE\* will not be automatically closed when the stdio\_filebuf is closed/destroyed.

References [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_buf\\_size](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_mode](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_reading](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::\\_M\\_set\\_buffer\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::is\\_open\(\)](#).

### ~stdio\_filebuf()

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf () [virtual]
```

Closes the external data stream if the file descriptor constructor was used.

## 6.930.3 Member Function Documentation

### \_M\_create\_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_create_pback () [inline], [protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back. Referenced by [pbackfail\(\)](#).

### \_M\_destroy\_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback () throw () [inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward. Referenced by [overflow\(\)](#), [seekoff\(\)](#), [seekpos\(\)](#), [underflow\(\)](#), and [xsgetn\(\)](#).

### \_M\_set\_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (
 streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Referenced by [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

**close()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
() [inherited]
```

Closes the currently associated file.

**Returns**

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

References `basic_filebuf()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, and `is_open()`.

Referenced by `std::basic_filebuf< char_type, traits_type >::~~basic_filebuf()`, and `open()`.

**eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**egptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char_type, traits_type, _Alloc >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

**epptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**fd()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf< _CharT, _Traits >::fd () [inline]
```

**Returns**

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

**file()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_filebuf< _CharT, _Traits >::file () [inline]
```

**Returns**

The underlying FILE\*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::ebackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::ebackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputbackc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::uflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc() const [inline], [inherited]
Locale access.
```

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubimbue()`.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::gptr() const [inline], [protected], [inherited]
Access to the get area.
```

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf<char_type, traits_type>::_M_create_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy`, `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_streambuf<char_type, traits_type>::in_avail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_streambuf<char_type, traits_type>::seekoff()`, `std::basic_streambuf<char_type, traits_type>::sgetc()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<char_type, traits_type>::showmanyc()`, `std::basic_streambuf<char_type, traits_type>::sungetc()`, `std::basic_streambuf<char_type, traits_type>::uflow()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::underflow()`, `std::basic_filebuf<_CharT, _Traits>::xsgetn()`, and `xsgetn()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::imbue (
 const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_ext_buf`, `_M_ext_next`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `is_open()`, and `seekoff()`.

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**is\_open()**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw () [inline], [nodiscard],
[inherited]
```

Returns true if the external file is open.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, `close()`, `imbue()`, `open()`, `seekoff()`, `seekpos()`, `setbuf()`, and `showmanyc()`.

**open() [1/3]**

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
 const _Path & __s,
 ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| <code>__s</code>    | The name of the file, as a <code>filesystem::path</code> . |
| <code>__mode</code> | The open mode flags.                                       |

**Returns**

`this` on success, `NULL` on failure

**open() [2/3]**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
 const char * __s,
 ios_base::openmode __mode) [inherited]
```

Opens an external file.

**Parameters**

|                  |                       |
|------------------|-----------------------|
| <code>__s</code> | The name of the file. |
|------------------|-----------------------|



|                     |                      |
|---------------------|----------------------|
| <code>__mode</code> | The open mode flags. |
|---------------------|----------------------|

**Returns**

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
| -----                     |    |     |       |     |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
| -----                     |    |     |       |     |                  |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |
| -----                     |    |     |       |     |                  |

References [\\_M\\_mode](#), [\\_M\\_reading](#), [\\_M\\_set\\_buffer\(\)](#), [std::ios\\_base::ate](#), [close\(\)](#), [std::ios\\_base::end](#), [is\\_open\(\)](#), and [seekoff\(\)](#).

Referenced by [std::basic\\_filebuf<char\\_type, traits\\_type>::open\(\)](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>::open\(\)](#).

**open() [3/3]**

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open (
 const std::string & __s,
 ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, NULL on failure

**overflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::app`, `std::ios_base::cur`, `std::ios_base::out`, `std::basic_streambuf< _CharT, _Traits >::pbase()`, `std::basic_streambuf< _CharT, _Traits >::pbackfail()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

Referenced by `pbackfail()`, `sync()`, `underflow()`, and `xsgetn()`.

**pbackfail()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_create_pback()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::in`, `overflow()`, `seekoff()`, and `underflow()`.

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, and `std::basic_streambuf<char_type, traits_type>::sputc()`.

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_streambuf<char_type, traits_type>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_filebuf<_CharT, _Traits>::xsputn()`, and `xsputn()`.

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__sp</code>   | Position                      |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekpos function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
```

```
char_type * __s,
streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

### **pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

### **sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]
```

Getting the next character.

#### **Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

### **seekoff()**

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### **Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_destroy_pback()`, `_M_reading`, `std::ios_base::cur`, `is_open()`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

Referenced by `imbue()`, `open()`, and `pbackfail()`.

### **seekpos()**

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_destroy_pback()`, `std::ios_base::beg`, and `is_open()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >::↵
setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html↵  
#io.streambuf.buffering](https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering) for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf`, `_M_buf_size`, and `is_open()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_filebuf<char_type, traits_type>::_M_create()`, `std::basic_filebuf<char_type, traits_type>::_M_destroy_pback()`, `std::basic_filebuf<char_type, traits_type>::_M_set_buffer()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, and `std::basic_filebuf<char_type, traits_type>::_M_s`

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ign` and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc () [protected], [virtual], [inherited]
Investigating the data available.
```

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

References `_M_mode`, `std::ios_base::binary`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::ios_base::in`, and `is_open()`.

**snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
Getting the next character.
```

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, char_traits< _CharT > >::get()`, and `std::basic_istream< _CharT, char_traits< _CharT > >::readsome()`.

**sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.



**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by [std::basic\\_istream< \\_CharT, \\_Traits >::unget\(\)](#).

**sync()**

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync () [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

References [overflow\(\)](#), [std::basic\\_streambuf< \\_CharT, \\_Traits >::pbase\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::pptr\(\)](#).

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), and [xsgetn\(\)](#).

**underflow()**

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::ios_base::in`, and `overflow()`.

Referenced by `pbackfail()`.

**xsggetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::xsggetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_destroy_pback()`, `_M_mode`, `_M_pback_init`, `_M_reading`, `_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gbump()`, `std::basic_streambuf<_CharT, _Traits>::in`, `std::ios_base::in`, `overflow()`, `std::basic_streambuf<_CharT, _Traits>::setg()`, and `std::basic_streambuf<char_type, traits_type>::xsggetn()`.

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

References `_M_buf_size`, `_M_mode`, `_M_reading`, `_M_set_buffer()`, `std::ios_base::app`, `std::basic_streambuf<_CharT, _Traits>::eptr()`, `std::ios_base::out`, `std::basic_streambuf<_CharT, _Traits>::pbase()`, `std::basic_streambuf<_CharT, _Traits>::pptr()`, and `std::basic_streambuf<char_type, traits_type>::xspn()`.

**6.930.4 Member Data Documentation****`_M_buf`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf<_CharT, _Traits>::_M_buf [protected], [inherited]
```

Pointer to the beginning of internal buffer.

Referenced by `basic_filebuf()`, and `setbuf()`.

**`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

**`_M_buf_size`**

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf<_CharT, _Traits>::_M_buf_size [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Referenced by `basic_filebuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::overflow()`, `setbuf()`, `underflow()`, `xsgn()`, and `xspn()`.

**`_M_ext_buf`**

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf<_CharT, _Traits>::_M_ext_buf [protected], [inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Referenced by `basic_filebuf()`, and `imbue()`.

**`_M_ext_buf_size`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size [protected], [inherited]
```

Size of buffer held by `_M_ext_buf`.

Referenced by [basic\\_filebuf\(\)](#).

### `_M_ext_next`

```
template<typename _CharT, typename _Traits>
```

```
const char* std::basic_filebuf<_CharT, _Traits>::_M_ext_next [protected], [inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Referenced by [basic\\_filebuf\(\)](#), and [imbue\(\)](#).

### `_M_in_beg`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg [protected], [inherited]
```

Start of get area.

### `_M_in_cur`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur [protected], [inherited]
```

Current read area.

### `_M_in_end`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end [protected], [inherited]
```

End of get area.

### `_M_mode`

```
template<typename _CharT, typename _Traits>
```

```
ios_base::openmode std::basic_filebuf<_CharT, _Traits>::_M_mode [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [showmanyc\(\)](#), [underflow\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

### `_M_out_beg`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

### `_M_out_cur`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
```

Current put area.

### `_M_out_end`

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_end [protected], [inherited]
```

End of put area.

### **`_M_pback`**

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected], [inherited]
```

Necessary bits for putback buffer management.

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_cur_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected], [inherited]
```

Necessary bits for putback buffer management.

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_end_save`**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected], [inherited]
```

Necessary bits for putback buffer management.

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#).

### **`_M_pback_init`**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected], [inherited]
```

Necessary bits for putback buffer management.

#### **Note**

pbacks of over one character are not currently supported.

Referenced by [basic\\_filebuf\(\)](#), [close\(\)](#), [pbackfail\(\)](#), and [xsgetn\(\)](#).

### **`_M_reading`**

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected], [inherited]
_M_reading == false && _M_writing == false for uncommitted mode; _M_reading == true for read mode; _M_writing
== true for write mode;
```

NB: `_M_reading == true && _M_writing == true` is unused.

Referenced by [basic\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::stdio\\_filebuf\(\)](#), [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >::close\(\)](#), [imbue\(\)](#), [open\(\)](#), [overflow\(\)](#), [pbackfail\(\)](#), [seekoff\(\)](#), [xsgetn\(\)](#), and [xsputn\(\)](#).

The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

**6.931 `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>` Class Template Reference**

```
#include <stdio_sync_filebuf.h>
```

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:

**Public Types**

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `_Traits` **traits\_type**

**Public Member Functions**

- **stdio\_sync\_filebuf** (`std::__c_file *__f`)
- **stdio\_sync\_filebuf** (`stdio_sync_filebuf &&__fb`) noexcept
- `std::__c_file * file` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- `stdio_sync_filebuf & operator=` (`stdio_sync_filebuf &&__fb`) noexcept
- locale `pubimbue` (const locale &\_\_loc)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- streamsize `sgetn` (`char_type *__s`, streamsize \_\_n)
- `int_type snextc` ()
- `int_type sputbackc` (`char_type __c`)
- `int_type sputc` (`char_type __c`)
- streamsize `sputn` (const `char_type *__s`, streamsize \_\_n)
- `int_type sungetc` ()
- void **swap** (`stdio_sync_filebuf &__fb`)
- `basic_streambuf * pubsetbuf` (`char_type *__s`, streamsize \_\_n)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
- void **\_\_safe\_pbump** (streamsize \_\_n)
- void **gbump** (int \_\_n)
- virtual void **imbue** (const locale &\_\_loc)
- virtual int\_type **overflow** (int\_type \_\_c=traits\_type::eof())
- virtual int\_type **pbackfail** (int\_type \_\_c=traits\_type::eof())
- void **pbump** (int \_\_n)
- virtual **std::streampos seekoff** (**std::streamoff** \_\_off, **std::ios\_base::seekdir** \_\_dir, **std::ios\_base::openmode**=**std::ios\_base::in**|**std::ios\_base::out**)
- virtual pos\_type **seekoff** (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual **std::streampos seekpos** (**std::streampos** \_\_pos, **std::ios\_base::openmode** \_\_mode=**std::ios\_base::in**|**std::ios\_base::out**)
- virtual pos\_type **seekpos** (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual **basic\_streambuf**< char\_type, \_Traits > \* **setbuf** (char\_type \*, streamsize)
- void **setg** (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
- void **setp** (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
- virtual streamsize **showmanyc** ()
- void **swap** (**basic\_streambuf** &\_\_sb)
- virtual int **sync** ()
- **stdio\_sync\_filebuf**< wchar\_t >::int\_type **syncgetc** ()
- **stdio\_sync\_filebuf**< char >::int\_type **syncgetc** ()
- int\_type **syncgetc** ()
- **stdio\_sync\_filebuf**< wchar\_t >::int\_type **syncputc** (int\_type \_\_c)
- **stdio\_sync\_filebuf**< char >::int\_type **syncputc** (int\_type \_\_c)
- int\_type **syncputc** (int\_type \_\_c)
- **stdio\_sync\_filebuf**< wchar\_t >::int\_type **syncungetc** (int\_type \_\_c)
- **stdio\_sync\_filebuf**< char >::int\_type **syncungetc** (int\_type \_\_c)
- int\_type **syncungetc** (int\_type \_\_c)
- virtual int\_type **uflow** ()
- virtual int\_type **underflow** ()
- **std::streamsize xsgetn** (char \*\_\_s, **std::streamsize** \_\_n)
- virtual **std::streamsize xsgetn** (char\_type \*\_\_s, **std::streamsize** \_\_n)
- **std::streamsize xsgetn** (wchar\_t \*\_\_s, **std::streamsize** \_\_n)
- virtual streamsize **xsgetn** (char\_type \*\_\_s, streamsize \_\_n)
- **std::streamsize xspn** (const char \*\_\_s, **std::streamsize** \_\_n)
- virtual **std::streamsize xspn** (const char\_type \*\_\_s, **std::streamsize** \_\_n)
- **std::streamsize xspn** (const wchar\_t \*\_\_s, **std::streamsize** \_\_n)
- virtual streamsize **xspn** (const char\_type \*\_\_s, streamsize \_\_n)
  
- char\_type \* **eback** () const
- char\_type \* **gptr** () const
- char\_type \* **egptr** () const
  
- char\_type \* **pbase** () const
- char\_type \* **pptr** () const
- char\_type \* **epptr** () const



## Protected Attributes

- locale [\\_M\\_buf\\_locale](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)

### 6.931.1 Detailed Description

**template**<typename `_CharT`, typename `_Traits` = `std::char_traits<_CharT>`>>  
**class** `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

### 6.931.2 Member Function Documentation

#### **eback()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::eback\(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_destroy\\_pback\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::imbue\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::pbackfail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_streambuf<char\\_type, traits\\_type>::sputbackc\(\)](#), [std::basic\\_streambuf<char\\_type, traits\\_type>::sungetc\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::underflow\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#).

#### **egptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits>::egptr\(\) const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by [std::basic\\_filebuf<char\\_type, traits\\_type>::\\_M\\_create\\_pback\(\)](#), [std::basic\\_streambuf<char\\_type, traits\\_type>::in\\_avail\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::overflow\(\)](#), [std::basic\\_streambuf<char\\_type, traits\\_type>::sbumpc\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>::seekpos\(\)](#), [std::basic\\_streambuf<char\\_type, traits\\_type>::sgetc\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::showmanyc\(\)](#), [std::basic\\_stringbuf<char\\_type, traits\\_type>::underflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>::underflow\(\)](#), [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>::underflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits>::xsgetn\(\)](#), and [xsgetn\(\)](#).

**epptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekpos\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**file()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file () [inline]
```

**Returns**

The underlying FILE\*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|       |                             |
|-------|-----------------------------|
| $\_n$ | The delta by which to move. |
|-------|-----------------------------|

This just advances the read position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::pbackfail\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::pbackfail\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sbumpc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sputbackc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::sungetc\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::uflow\(\)](#), and [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsgetn\(\)](#).

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by [std::basic\\_streambuf< char\\_type, traits\\_type >::pubimbue\(\)](#).

## gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

## imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

### Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

### Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_filebuf< wchar_t >`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

## in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::overflow (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**pbackfail()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, and `std::basic_filebuf<_CharT, _Traits>::xsputn()`.

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, and `std::basic_streambuf<char_type, traits_type>::sputc()`.

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Referenced by `std::basic_filebuf<_CharT, _Traits>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_filebuf<_CharT, _Traits>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_streambuf<char_type, traits_type>::str()`, `std::basic_filebuf<_CharT, _Traits>::sync()`, `std::basic_filebuf<_CharT, _Traits>::xsputn()`, and `xsputn()`.

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived `imbue(__loc)`.

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekoff` function.

**pubseekpos()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

**Parameters**

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

**pubsetbuf()**

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

**pubsync()**

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

**sbumpc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_iostream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**seekoff()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekoff()`.

**seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubseekpos()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
 char_type * ,
 streamsize) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, and `std::basic_stringbuf< char >`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubsetbuf()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert()`, `std::basic_filebuf< char_type, traits_type >::_M_create()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_set_buffer()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.



**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, and `std::basic_filebuf<char_type, traits_type>::_M_s`

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sgetc () [inline], [inherited]
Getting the next character.
```

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_iostream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_streambuf<char_type, traits_type>::snextc()`.

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected],
[virtual], [inherited]
Investigating the data available.
```

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_filebuf<wchar_t>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_stringbuf<char>`, and `std::basic_stringbuf<wchar_t>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::in_avail()`.

## snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Referenced by `std::basic_istream<char_type, traits_type>::get()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, char_traits<_CharT>>::get()`, and `std::basic_istream<_CharT, char_traits<_CharT>>::readsome()`.

## sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

## Returns

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

**sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**sync()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::sync () [inline], [protected],
[virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**uflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::uflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**underflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::underflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

**xsggetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsggetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, and `std::basic_filebuf<char_type, traits_type>`.

References `egptr()`, `gptr()`, `std::min()`, and `uflow()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sgetn()`.

**xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, and `std::basic_filebuf<char_type, traits_type>`.

References `epptr()`, `std::min()`, `overflow()`, and `pptr()`.

Referenced by `std::basic_streambuf<char_type, traits_type>::sputn()`.

### 6.931.3 Member Data Documentation

#### **`_M_buf_locale`**

```
template<typename _CharT, typename _Traits>
```

```
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf](#)< \_CharT, \_Traits >::basic\_filebuf().

#### **`_M_in_beg`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

#### **`_M_in_cur`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

#### **`_M_in_end`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

#### **`_M_out_beg`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

#### **`_M_out_cur`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

#### **`_M_out_end`**

```
template<typename _CharT, typename _Traits>
```

```
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

### 6.932 `std::chrono::steady_clock` Struct Reference

```
#include <chrono.h>
```

#### **Public Types**

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**

- typedef `duration::rep` **rep**
- typedef `chrono::time_point< steady_clock, duration >` **time\_point**

#### Static Public Member Functions

- static `time_point now ()` noexcept

#### Static Public Attributes

- static constexpr bool **is\_steady**

#### 6.932.1 Detailed Description

Monotonic clock.

Time returned has the property of only increasing at a uniform rate.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

### 6.933 `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >`:



#### Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

#### Public Attributes

- `hash_type` **m\_hash**
- `value_type` **m\_value**

### 6.933.1 Detailed Description

```
template<typename _Tv, typename _Th, bool Store_Hash>
struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.  
The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

### 6.934 \_\_gnu\_pbds::detail::stored\_data< \_Tv, \_Th, false > Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::stored\_data< \_Tv, \_Th, false >:



#### Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

#### Public Attributes

- hash\_type **m\_hash**
- value\_type **m\_value**

### 6.934.1 Detailed Description

```
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >
```

Specialization for representation of stored data of just value type.  
The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)



**6.935 `__gnu_pbds::detail::stored_hash<_Th>` Struct Template Reference**

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:

**Public Types**

- typedef `_Th` **hash\_type**

**Public Attributes**

- hash\_type **m\_hash**

**6.935.1 Detailed Description**

```
template<typename _Th>
struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

**6.936 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference**

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::stored_value<_Tv>`:



### Public Types

- `typedef _Tv value_type`

### Public Attributes

- `value_type m_value`

#### 6.936.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 6.937 \_\_gnu\_pbds::string\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::string_tag`:



### 6.937.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.938 `std::student_t_distribution<_RealType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

### Public Member Functions

- **student\_t\_distribution** (`_RealType __n`)
- **student\_t\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator`>  
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator`>  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng`, const [param\\_type](#) &\_\_p)

- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator>  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator>  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::student_t_distribution< _RealType1 > &__x)`
- `bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::student_t_distribution< _RealType1 > &__x)`

### 6.938.1 Detailed Description

`template<typename _RealType = double>  
class std::student_t_distribution< _RealType >`

A `student_t_distribution` random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Since

C++11

### 6.938.2 Member Typedef Documentation

#### result\_type

`template<typename _RealType = double>  
typedef _RealType std::student_t_distribution< _RealType >::result_type`  
The type of the range of the distribution.

### 6.938.3 Member Function Documentation

#### max()

`template<typename _RealType = double>  
result_type std::student_t_distribution< _RealType >::max () const [inline]`  
Returns the least upper bound value of the distribution.

#### min()

`template<typename _RealType = double>  
result_type std::student_t_distribution< _RealType >::min () const [inline]`  
Returns the greatest lower bound value of the distribution.

**operator()()**

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::student_t_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

**param() [1/2]**

```
template<typename _RealType = double>
param_type std::student_t_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

**param() [2/2]**

```
template<typename _RealType = double>
void std::student_t_distribution<_RealType>::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::student_t_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

**6.938.4 Friends And Related Symbol Documentation****operator<<**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream<_CharT, _Traits> & operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::student_t_distribution<_RealType1> & __x) [friend]
```

Inserts a student\_t\_distribution random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <code>__os</code> | An output stream.                                    |
| <code>__x</code>  | A student_t_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**operator==**

```
template<typename _RealType = double>
bool operator== (
 const student_t_distribution< _RealType > & __d1,
 const student_t_distribution< _RealType > & __d2) [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

**operator>>**

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::student_t_distribution< _RealType1 > & __x) [friend]
```

Extracts a student\_t\_distribution random number distribution \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <code>__is</code> | An input stream.                                         |
| <code>__x</code>  | A student_t_distribution random number generator engine. |

**Returns**

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.939 std::sub\_match<\_Bilter > Class Template Reference**

```
#include <regex>
```

**Public Types**

- typedef \_\_iter\_traits::difference\_type **difference\_type**
- typedef \_Bilter **iterator**
- typedef basic\_string< value\_type > **string\_type**
- typedef \_\_iter\_traits::value\_type **value\_type**

**Public Member Functions**

- int [compare](#) (const sub\_match &\_\_s) const
- difference\_type [length](#) () const noexcept
- [operator string\\_type](#) () const
- string\_type [str](#) () const
- void [swap](#) (sub\_match &\_\_s) noexcept(\_\_is\_nothrow\_swappable< \_Bilter >::value)
- int [compare](#) (const string\_type &\_\_s) const
- int [compare](#) (const value\_type \*\_\_s) const

## Public Attributes

- iterator **first**
- bool **matched**
- iterator **second**

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Bilter>`  
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter>`  
`auto operator<=> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bilter >::value)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Bi_iter>`  
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter>`  
`auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>`  
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`

### 6.939.1 Detailed Description

**template<typename \_Bilter>**  
**class std::sub\_match<\_Bilter >**

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of the same character type as the pattern matched by the regular expression.

A `sub_match<Iter>` has a public base class of type `pair<Iter, Iter>`, so inherits `pair`'s data members named `first` and `second`. The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Since

C++11

## 6.939.2 Member Function Documentation

### compare() [1/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
 const string_type & __s) const [inline]
```

Compares this sub\_match to a string.

#### Parameters

|       |                                        |
|-------|----------------------------------------|
| $\_s$ | A string to compare to this sub_match. |
|-------|----------------------------------------|

#### Return values

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before $\_s$ . |
| <i>zero</i>     | This matched sequence is equivalent to $\_s$ .    |
| <i>positive</i> | This matched sequence will collate after $\_s$ .  |

### compare() [2/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
 const sub_match< _BiIter > & __s) const [inline]
```

Compares this and another matched sequence.

#### Parameters

|       |                                                  |
|-------|--------------------------------------------------|
| $\_s$ | Another matched sequence to compare to this one. |
|-------|--------------------------------------------------|

#### Return values

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before $\_s$ . |
| <i>zero</i>     | This matched sequence is equivalent to $\_s$ .    |
| <i>positive</i> | This matched sequence will collate after $\_s$ .  |

Referenced by `std::sub_match< const char * >::operator==(, and std::sub_match< const char * >::operator==(.`

### compare() [3/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
 const value_type * __s) const [inline]
```

Compares this sub\_match to a string.



**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↔</code> | A string to compare to this <code>sub_match</code> . |
| <code>_s</code> |                                                      |

**Return values**

|                 |                                                              |
|-----------------|--------------------------------------------------------------|
| <i>negative</i> | This matched sequence will collate before <code>__s</code> . |
| <i>zero</i>     | This matched sequence is equivalent to <code>__s</code> .    |
| <i>positive</i> | This matched sequence will collate after <code>__s</code> .  |

**length()**

```
template<typename _Biter>
difference_type std::sub_match<_Biter>::length() const [inline], [noexcept]
```

Gets the length of the matching sequence.

**operator string\_type()**

```
template<typename _Biter>
std::sub_match<_Biter>::operator string_type() const [inline]
```

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

**str()**

```
template<typename _Biter>
string_type std::sub_match<_Biter>::str() const [inline]
```

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

Referenced by `std::sub_match<const char*>::operator string_type()`.

**swap()**

```
template<typename _Biter>
void std::sub_match<_Biter>::swap (
 sub_match<_Biter> & __s) [inline], [noexcept]
```

Swap the values of two `sub_match` objects.

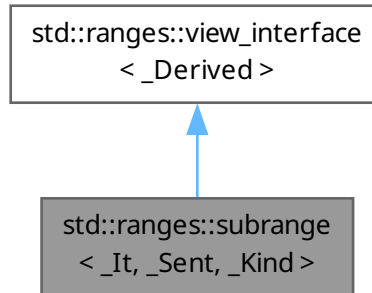
The documentation for this class was generated from the following file:

- [regex.h](#)

## 6.940 std::ranges::subrange<\_It, \_Sent, \_Kind> Class Template Reference

```
#include <ranges_util.h>
```

Inheritance diagram for std::ranges::subrange<\_It, \_Sent, \_Kind>:



### Public Member Functions

- constexpr **subrange** (\_\_detail::\_\_convertible\_to\_non\_slicing<\_It> auto \_\_i, \_Sent \_\_s) noexcept(is\_nothrow\_constructible\_v<\_It, decltype(\_\_i)> &&is\_nothrow\_constructible\_v<\_Sent, \_Sent &>)
- constexpr **subrange** (\_\_detail::\_\_convertible\_to\_non\_slicing<\_It> auto \_\_i, \_Sent \_\_s, \_\_size\_type \_\_n) noexcept(is\_nothrow\_constructible\_v<\_It, decltype(\_\_i)> &&is\_nothrow\_constructible\_v<\_Sent, \_Sent &>)
- template<\_\_detail::\_\_different\_from<subrange> \_Rng>  
requires \_S\_store\_size  
constexpr **subrange** (\_Rng &&\_\_r) noexcept(noexcept(subrange(\_\_r, ranges::size(\_\_r)))) &&sized\_range<\_Rng>
- template<\_\_detail::\_\_different\_from<subrange> \_Rng>  
requires (!\_S\_store\_size)  
constexpr **subrange** (\_Rng &&\_\_r) noexcept(noexcept(subrange(ranges::begin(\_\_r), ranges::end(\_\_r))))
- template<borrowed\_range \_Rng>  
requires (\_Kind == subrange\_kind::sized)  
constexpr **subrange** (\_Rng &&\_\_r, \_\_size\_type \_\_n) noexcept(noexcept(subrange(ranges::begin(\_\_r), ranges::end(\_\_r), \_\_n)))
- constexpr subrange & **advance** (iter\_difference\_t<\_It> \_\_n)
- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr \_It **begin** ()
- constexpr \_It **begin** () const
- constexpr auto **cbegin** ()
- constexpr auto **cbegin** () const
- constexpr auto **cend** ()
- constexpr auto **cend** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr bool **empty** () const
- constexpr bool **empty** () noexcept(noexcept(\_S\_empty(\_M\_derived())))

- constexpr `_Sent` **end** () const
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr subrange **next** (iter\_difference\_t<\_It> \_\_n=1) &&
- constexpr subrange **next** (iter\_difference\_t<\_It> \_\_n=1) const &
- template<\_\_detail::\_\_different\_from<subrange> \_PairLike>  
constexpr **operator** `_PairLike` () const
- constexpr **operator** `bool` () const noexcept(noexcept(ranges::empty(\_M\_derived())))
- constexpr **operator** `bool` () noexcept(noexcept(ranges::empty(\_M\_derived())))
- template<random\_access\_range \_Range = \_Derived>  
constexpr decltype(auto) **operator**[] (range\_difference\_t<\_Range> \_\_n)
- template<random\_access\_range \_Range = const \_Derived>  
constexpr decltype(auto) **operator**[] (range\_difference\_t<\_Range> \_\_n) const
- constexpr subrange **prev** (iter\_difference\_t<\_It> \_\_n=1) const
- constexpr \_\_size\_type **size** () const
- constexpr auto **size** () noexcept(noexcept(\_S\_size(\_M\_derived())))

## Friends

- struct `views::_Drop`

### 6.940.1 Detailed Description

```
template<input_or_output_iterator _It, sentinel_for<_It> _Sent = _It, subrange_kind _Kind = sized_sentinel,
 _for<_Sent, _It> ? subrange_kind::sized : subrange_kind::unsized>
requires (_Kind == subrange_kind::sized || !sized_sentinel_for<_Sent, _It>)
class std::ranges::subrange<_It, _Sent, _Kind>
```

The `ranges::subrange` class template.

The documentation for this class was generated from the following file:

- [ranges\\_util.h](#)

## 6.941 `std::subtract_with_carry_engine<_UIntType, __w, __s, __r>` Class Template Reference

```
#include <random>
```

## Public Types

- typedef `_UIntType` [result\\_type](#)

## Public Member Functions

- template<typename \_Sseq, typename = \_If\_seed\_seq<\_Sseq>>  
[subtract\\_with\\_carry\\_engine](#) (\_Sseq &\_\_q)
- [subtract\\_with\\_carry\\_engine](#) ([result\\_type](#) \_\_sd)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- template<typename \_Sseq>  
\_If\_seed\_seq<\_Sseq> > [seed](#) (\_Sseq &\_\_q)
- void [seed](#) ([result\\_type](#) \_\_sd=0u)

### Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

### Static Public Attributes

- static constexpr uint\_least32\_t **default\_seed**
- static constexpr size\_t **long\_lag**
- static constexpr size\_t **short\_lag**
- static constexpr size\_t **word\_size**

### Friends

- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)
- bool [operator==](#) (const [subtract\\_with\\_carry\\_engine](#) &\_\_lhs, const [subtract\\_with\\_carry\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::subtract\\_with\\_carry\\_engine](#)< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)

#### 6.941.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >
```

The Marsaglia-Zaman generator.

This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.

A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is  $r$  and the maximum period of the generator is  $(m^r - m^s - 1)$ .

Since

C++11

#### 6.941.2 Member Typedef Documentation

##### result\_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

#### 6.941.3 Constructor & Destructor Documentation

##### subtract\_with\_carry\_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
 result_type __sd) [inline], [explicit]
```

Constructs an explicitly seeded [subtract\\_with\\_carry\\_engine](#) random number generator.

**subtract\_with\_carry\_engine()** [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
 _Sseq & __q) [inline], [explicit]
```

Constructs a subtract\_with\_carry\_engine random number engine seeded from the seed sequence \_\_q.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| $\leftrightarrow$ | the seed sequence. |
| <i>q</i>          |                    |

**6.941.4 Member Function Documentation****discard()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

**max()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::max () [inline],
[static], [constexpr]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

**min()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::min () [inline],
[static], [constexpr]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

**operator>()**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type std::subtract_with_carry_engine<
 _UIntType, __w, __s, __r >::operator() ()
```

Gets the next random number in the sequence.

**seed()** [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq>
auto std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
 _Sseq & __q) -> _If_seed_seq< _Sseq >
```

Seeds the initial state  $x_0$  of the % subtract\_with\_carry\_engine random number generator.

**seed()** [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
```

```
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
 result_type __sd = 0u)
```

Seeds the initial state  $x_0$  of the random number generator.

N1688[4.19] modifies this as follows. If `__value == 0`, sets value to 19780503. In any case, with a linear congruential generator  $lcg(i)$  having parameters  $m_{lcg} = 2147483563$ ,  $a_{lcg} = 40014$ ,  $c_{lcg} = 0$ , and  $lcg(0) = value$ , sets  $x_{-r} \dots x_{-1}$  to  $lcg(1) \bmod m \dots lcg(r) \bmod m$  respectively. If  $x_{-1} = 0$  set carry to 1, otherwise sets carry to 0.

Referenced by `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine()`, and `std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine()`.

### 6.941.5 Friends And Related Symbol Documentation

#### operator<<

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]
```

Inserts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                           |
| <code>__x</code>  | A % <code>subtract_with_carry_engine</code> random number generator engine. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### operator==

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
 const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
 const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs) [friend]
```

Compares two % `subtract_with_carry_engine` random number generator objects of the same type for equality.

#### Parameters

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| <code>__lhs</code> | A % <code>subtract_with_carry_engine</code> random number generator object.       |
| <code>__rhs</code> | Another % <code>subtract_with_carry_engine</code> random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

**operator>>**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]
Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.
```

**Parameters**

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <code>__is</code> | An input stream.                                               |
| <code>__x</code>  | A % subtract_with_carry_engine random number generator engine. |

**Returns**

The input stream with the state of `__x` extracted or in an error state.

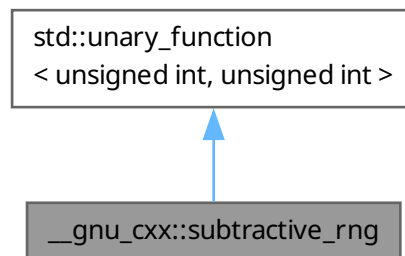
The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.942 `__gnu_cxx::subtractive_rng` Class Reference**

```
#include <functional>
```

Inheritance diagram for `__gnu_cxx::subtractive_rng`:

**Public Types**

- typedef unsigned int [argument\\_type](#)
- typedef unsigned int [result\\_type](#)

## Public Member Functions

- [subtractive\\_rng](#) ()
- [subtractive\\_rng](#) (unsigned int \_\_seed)
- void [\\_M\\_initialize](#) (unsigned int \_\_seed)
- unsigned int [operator\(\)](#) (unsigned int \_\_limit)

### 6.942.1 Detailed Description

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

### 6.942.2 Member Typedef Documentation

#### `argument_type`

typedef unsigned int [std::unary\\_function](#)< unsigned int, unsigned int >::argument\_type [inherited]  
`argument_type` is the type of the argument

#### `result_type`

typedef unsigned int [std::unary\\_function](#)< unsigned int, unsigned int >::result\_type [inherited]  
`result_type` is the return type

### 6.942.3 Constructor & Destructor Documentation

#### `subtractive_rng()` [1/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng (
 unsigned int __seed) [inline]
```

Ctor allowing you to initialize the seed.

#### `subtractive_rng()` [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng () [inline]
```

Default ctor; initializes its state with some number you don't see.

### 6.942.4 Member Function Documentation

#### `operator()()`

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
 unsigned int __limit) [inline]
```

Returns a number less than the argument.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 6.943 `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct` Template Reference

```
#include <synth_access_traits.hpp>
```

## Public Types

- typedef `_ATraits` **base\_type**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `type_traits::const_reference` **const\_reference**



- typedef type\_traits::key\_const\_reference **key\_const\_reference**
- typedef Type\_Traits **type\_traits**

### Public Member Functions

- **synth\_access\_traits** (const base\_type &)
- bool **cmp\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **cmp\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=false) const
- bool **equal\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **equal\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=true) const

### Static Public Member Functions

- static key\_const\_reference **extract\_key** (const\_reference)

#### 6.943.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >
```

Synthetic element access traits.

The documentation for this struct was generated from the following file:

- [synth\\_access\\_traits.hpp](#)

## 6.944 std::chrono::system\_clock Struct Reference

```
#include <chrono.h>
```

### Public Types

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)< [system\\_clock](#), duration > **time\_point**

### Static Public Member Functions

- static time\_point **from\_time\_t** (std::time\_t \_\_t) noexcept
- static time\_point **now** () noexcept
- static std::time\_t **to\_time\_t** (const time\_point &\_\_t) noexcept

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 6.944.1 Detailed Description

System clock.

Time returned represents wall time from the system-wide clock.

The documentation for this struct was generated from the following file:

- [chrono.h](#)

## 6.945 std::system\_error Class Reference

```
#include <system_error>
```

Inheritance diagram for std::system\_error:



### Public Member Functions

- **system\_error** (const system\_error &)=default
- **system\_error** (error\_code \_\_ec, const char \*\_\_what)
- **system\_error** (error\_code \_\_ec, const string &\_\_what)
- **system\_error** (error\_code \_\_ec=error\_code())
- **system\_error** (int \_\_v, const error\_category &\_\_ecat)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const char \*\_\_what)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const string &\_\_what)
- const error\_code & **code** () const noexcept
- system\_error & **operator=** (const system\_error &)=default
- virtual const char \* **what** () const noexcept

### 6.945.1 Detailed Description

An exception type that includes an `error_code` value.

Typically used to report errors from the operating system and other low-level APIs.

Since

C++11

### 6.945.2 Member Function Documentation

#### what()

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

## 6.946 std::chrono::tai\_clock Class Reference

```
#include <chrono>
```

## Public Types

- using **duration**
- using **period**
- using **rep**
- using **time\_point**

## Static Public Member Functions

- `template<typename _Duration>`  
`static tai_time< common\_type\_t< _Duration, seconds > > from_utc (const utc_time< _Duration > &__t)`
- `static time_point now ()`
- `template<typename _Duration>`  
`static utc_time< common\_type\_t< _Duration, seconds > > to_utc (const tai_time< _Duration > &__t)`

## Static Public Attributes

- static constexpr bool **is\_steady**

### 6.946.1 Detailed Description

A clock that measures International Atomic Time.  
 The epoch is 1958-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

## 6.947 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

```
#include <memory>
```

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



## Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- [temporary\\_buffer](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- [~temporary\\_buffer](#) ()
- size\_type [\\_M\\_requested\\_size](#) () const
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [size](#) () const

## Protected Attributes

- struct std::\_Temporary\_buffer::\_Impl **\_M\_impl**
- size\_type **\_M\_original\_len**

### 6.947.1 Detailed Description

**template<class \_ForwardIterator, class \_Tp = typename std::iterator\_traits<\_ForwardIterator>::value\_type>**  
**struct \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >**

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

### 6.947.2 Constructor & Destructor Documentation

#### `temporary_buffer()`

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
```

```
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (
 _ForwardIterator __first,
 _ForwardIterator __last) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

References [std::\\_Temporary\\_buffer< \\_ForwardIterator, \\_Tp >::\\_Temporary\\_buffer\(\)](#), and [\\_\\_gnu\\_cxx::distance\(\)](#).

#### `~temporary_buffer()`

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
```

```
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::~~temporary_buffer () [inline]
```

Destroys objects and frees storage.

### 6.947.3 Member Function Documentation

#### `_M_requested_size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

#### `begin()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline], [inherited]
```

As per Table mumble.

Referenced by [\\_Temporary\\_buffer\(\)](#).

#### `end()`

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline], [inherited]
```

As per Table mumble.

Referenced by [\\_Temporary\\_buffer\(\)](#).

#### `size()`

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline], [inherited]
```

As per Table mumble.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

## 6.948 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <thin_heap_.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**

- typedef \_\_rebind\_a::const\_pointer **const\_pointer**
- typedef \_\_rebind\_a::const\_reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef base\_type::iterator **iterator**
- typedef base\_type::point\_const\_iterator **point\_const\_iterator**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef \_\_rebind\_a::pointer **pointer**
- typedef \_\_rebind\_a::reference **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef Value\_Type **value\_type**

### Public Member Functions

- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point\_iterator)
- template<typename Pred>  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- void **join** (thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred>  
void **split** (Pred, thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** (left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef base\_type::node **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef base\_type::node\_const\_pointer **node\_const\_pointer**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef base\_type::node\_pointer **node\_pointer**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- **thin\_heap** (const Cmp\_Fn &)
- **thin\_heap** (const thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It>  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)

- void **swap** (thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

#### Static Protected Member Functions

- static node\_pointer **parent** (node\_pointer)

#### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 6.948.1 Detailed Description

template<typename Value\_Type, typename Cmp\_Fn, typename \_Alloc>  
class \_\_gnu\_pbds::detail::thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc >

Thin heap.

See Tarjan and Kaplan.

The documentation for this class was generated from the following file:

- [thin\\_heap\\_.hpp](#)

## 6.949 \_\_gnu\_pbds::thin\_heap\_tag Struct Reference

#include <tag\_and\_trait.hpp>

Inheritance diagram for \_\_gnu\_pbds::thin\_heap\_tag:



#### 6.949.1 Detailed Description

Thin heap.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.950 std::thread Class Reference

```
#include <thread>
```

### Classes

- class [id](#)

### Public Types

- using [native\\_handle\\_type](#)

### Public Member Functions

- `template<typename _Callable, typename... _Args, typename = _Require<__not_same<_Callable>>>>`  
`thread (_Callable &&__f, _Args &&... __args)`
- `thread (const thread &)=delete`
- `thread (thread &&__t) noexcept`
- `void detach ()`
- `id get_id () const noexcept`
- `void join ()`
- `bool joinable () const noexcept`
- `native_handle_type native_handle ()`
- `thread & operator= (const thread &)=delete`
- `thread & operator= (thread &&__t) noexcept`
- `void swap (thread &__t) noexcept`

### Static Public Member Functions

- static unsigned int [hardware\\_concurrency \(\)](#) noexcept

### Related Symbols

(Note that these are not member symbols.)

- `bool operator== (thread::id __x, thread::id __y) noexcept`
- `void swap (thread &__x, thread &__y) noexcept`

#### 6.950.1 Detailed Description

A `std::thread` represents a new thread of execution.

The default constructor creates an object that does not own a thread. The `thread(F&&, Args&&...)` constructor invokes a callable in a new thread, and owns that new thread. A `std::thread` that owns a thread is *joinable*. Joining a thread waits for it to finish executing, which happens when the callable running in that thread returns.

A `std::thread` cannot be copied, but can be moved. Moving a joinable object transfers ownership of its thread to another object.

A joinable `std::thread` must be explicitly joined (or detached) before it is destroyed or assigned to. Attempting to destroy a joinable thread will terminate the whole process.

Since

C++11



### 6.950.2 Member Function Documentation

#### `native_handle()`

```
native_handle_type std::thread::native_handle () [inline]
```

#### Precondition

thread is joinable

The documentation for this class was generated from the following file:

- [std\\_thread.h](#)

## 6.951 `__gnu_cxx::throw_allocator_base< _Tp, _Cond >` Class Template Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`:



### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint=0)
- void **check** (size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up>  
void **destroy** (\_Up \*\_\_p)

- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept

### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

#### 6.951.1 Detailed Description

```
template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
```

Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.

Note: Deallocate not allowed to throw.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

#### 6.952 \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp > Struct Template Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >:



## Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef std::true\_type propagate_on_container_move_assignment`
- `typedef value_type & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- **`throw_allocator_limit`** (const throw\_allocator\_limit &) noexcept
- `template<typename _Tp1>`  
  **`throw_allocator_limit`** (const throw\_allocator\_limit<\_Tp1> &) noexcept
- pointer **`address`** (reference \_\_x) const noexcept
- pointer **`allocate`** (size\_type \_\_n, const void \* \_\_hint=0)
- void **`check`** (size\_type \_\_n)
- `map_alloc_type::iterator` **`check_allocated`** (void \*p, size\_t size)
- void **`check_allocated`** (pointer \_\_p, size\_type \_\_n)
- void **`check_constructed`** (size\_t label)
- `map_construct_type::iterator` **`check_constructed`** (void \*p)
- void **`construct`** (\_Up \* \_\_p, \_Args &&... \_\_args)
- void **`deallocate`** (pointer \_\_p, size\_type \_\_n)
- void **`destroy`** (\_Up \* \_\_p)
- void **`erase`** (void \*p, size\_t size)
- void **`erase_construct`** (void \*p)
- void **`insert`** (void \*p, size\_t size)
- void **`insert_construct`** (void \*p)
- size\_type **`max_size`** () const noexcept
- throw\_allocator\_limit & **`operator=`** (const throw\_allocator\_limit &)=default

## Static Public Member Functions

- static void **`check`** ()
- static size\_t & **`count`** ()
- static size\_t **`get_label`** ()
- static size\_t & **`limit`** ()
- static void **`set_label`** (size\_t l)
- static void **`set_limit`** (const size\_t \_\_l)
- static void **`throw_conditionally`** ()

### 6.952.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit<_Tp>
```

Allocator throwing via limit condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 6.953 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



#### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **throw\_allocator\_random** (const throw\_allocator\_random &) noexcept
- template<typename \_Tp1>  
  **throw\_allocator\_random** (const throw\_allocator\_random<\_Tp1> &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*\_\_hint=0)
- void **check** (size\_type \_\_n)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_constructed** (size\_t label)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)

- void **deallocate** (pointer `__p`, size\_type `__n`)
- void **destroy** (`_Up *``__p`)
- void **erase** (void `*p`, size\_t `size`)
- void **erase\_construct** (void `*p`)
- void **insert** (void `*p`, size\_t `size`)
- void **insert\_construct** (void `*p`)
- size\_type **max\_size** () const noexcept
- throw\_allocator\_random & **operator=** (const throw\_allocator\_random &)=default
- void **seed** (unsigned long `__s`)

### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t `l`)
- static void **set\_probability** (double `__p`)
- static void **throw\_conditionally** ()

#### 6.953.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_random<_Tp>
```

Allocator throwing via random condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 6.954 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

```
#include <throw_allocator.h>
```

### Public Types

- typedef `_Cond` **condition\_type**

### Public Member Functions

- **throw\_value\_base** (const std::size\_t `__i`)
- **throw\_value\_base** (const throw\_value\_base & `__v`)
- **throw\_value\_base** (throw\_value\_base &&)=default
- throw\_value\_base & **operator++** ()
- throw\_value\_base & **operator=** (const throw\_value\_base & `__v`)
- throw\_value\_base & **operator=** (throw\_value\_base &&)=default

### Public Attributes

- std::size\_t **\_M\_i**

### 6.954.1 Detailed Description

```
template<typename _Cond>
struct __gnu_cxx::throw_value_base< _Cond >
```

Class with exception generation control. Intended to be used as a value\_type in templated code.

Note: Destructor not allowed to throw.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 6.955 \_\_gnu\_cxx::throw\_value\_limit Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for \_\_gnu\_cxx::throw\_value\_limit:



#### Public Types

- typedef throw\_value\_base< [limit\\_condition](#) > **base\_type**
- typedef [limit\\_condition](#) **condition\_type**

#### Public Member Functions

- **throw\_value\_limit** (const std::size\_t \_\_i)
- **throw\_value\_limit** (const throw\_value\_limit &\_\_other)
- **throw\_value\_limit** (throw\_value\_limit &&)=default
- throw\_value\_base & **operator++** ()
- throw\_value\_limit & **operator=** (const throw\_value\_limit &\_\_other)
- throw\_value\_limit & **operator=** (throw\_value\_limit &&)=default

**Static Public Member Functions**

- static `size_t & count ()`
- static `size_t & limit ()`
- static void `set_limit (const size_t __l)`
- static void `throw_conditionally ()`

**Public Attributes**

- `std::size_t _M_i`

**6.955.1 Detailed Description**

Type throwing via limit condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**6.956 \_\_gnu\_cxx::throw\_value\_random Struct Reference**

```
#include <throw_allocator.h>
```

Inheritance diagram for `__gnu_cxx::throw_value_random`:

**Public Types**

- typedef `throw_value_base< random\_condition > base_type`
- typedef `random\_condition condition_type`

### Public Member Functions

- **throw\_value\_random** (const std::size\_t \_\_i)
- **throw\_value\_random** (const throw\_value\_random &\_\_other)
- **throw\_value\_random** (throw\_value\_random &&)=default
- throw\_value\_base & **operator++** ()
- throw\_value\_random & **operator=** (const throw\_value\_random &\_\_other)
- throw\_value\_random & **operator=** (throw\_value\_random &&)=default
- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

### Public Attributes

- std::size\_t **M\_i**

#### 6.956.1 Detailed Description

Type throwing via random condition.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 6.957 std::time\_base Class Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_base:





**Public Types**

- enum **dateorder** {  
    **no\_order** , **dmy** , **mdy** , **ymd** ,  
    **ydm** }

**6.957.1 Detailed Description**

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

**6.958 std::time\_get<\_CharT, \_InIter> Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_get<\_CharT, \_InIter>:

**Public Types**

- enum **dateorder** {  
    **no\_order** , **dmy** , **mdy** , **ymd** ,  
    **ydm** }
- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)

**Public Member Functions**

- [time\\_get](#) (size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const

- `iter_type get (iter_type __s, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, char __format, char __modifier=0) const`
- `iter_type get (iter_type __s, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, const char_type * __fmt, const char_type * __fmtend) const`
- `iter_type get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_time (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_weekday (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- `iter_type get_year (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`

### Static Public Attributes

- static `locale::id`

### Protected Member Functions

- virtual `~time_get ()`
- `iter_type M_extract_name (iter_type __beg, iter_type __end, int & __member, const _CharT ** __names, size_t __indexlen, ios_base & __io, ios_base::iostate & __err) const`
- `iter_type M_extract_num (iter_type __beg, iter_type __end, int & __member, int __min, int __max, size_t __len, ios_base & __io, ios_base::iostate & __err) const`
- `iter_type M_extract_via_format (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, const _CharT * __format) const`
- `iter_type M_extract_via_format (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm, const _CharT * __format, __time_get_state & __state) const`
- `iter_type M_extract_wday_or_month (iter_type __beg, iter_type __end, int & __member, const _CharT ** __names, size_t __indexlen, ios_base & __io, ios_base::iostate & __err) const`
- virtual dateorder `do_date_order ()` const
- `iter_type do_get (iter_type __s, iter_type __end, ios_base & __f, ios_base::iostate & __err, tm * __tm, char __format, char __modifier) const`
- virtual `iter_type do_get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_time (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
- virtual `iter_type do_get_year (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`

### Static Protected Member Functions

- static `_c_locale _S_clone_c_locale (_c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (_c_locale & __cloc, const char * __s, _c_locale __old=0)`
- static void `_S_destroy_c_locale (_c_locale & __cloc)`
- static `_c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `_c_locale _S_lc_ctype_c_locale (_c_locale __cloc, const char * __s)`

### 6.958.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get< _CharT, _InIter >
```

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The `time_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_get` facet.

### 6.958.2 Member Typedef Documentation

#### char\_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::time_get< _CharT, _InIter >::char_type
Public typedefs.
```

#### iter\_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
Public typedefs.
```

### 6.958.3 Constructor & Destructor Documentation

#### time\_get()

```
template<typename _CharT, typename _InIter>
std::time_get< _CharT, _InIter >::time_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

#### ~time\_get()

```
template<typename _CharT, typename _InIter>
virtual std::time_get< _CharT, _InIter >::~~time_get () [inline], [protected], [virtual]
Destructor.
```

### 6.958.4 Member Function Documentation

#### date\_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const [inline]
Return preferred order of month, day, and year.
```

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_↵put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `time_base::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

#### Returns

A member of `time_base::dateorder`.

References [do\\_date\\_order\(\)](#).

#### do\_date\_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order () const [protected], [virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_↵put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of `time_base::dateorder`.

Referenced by [date\\_order\(\)](#).

#### do\_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 ios_base & __f,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier) const [inline], [protected]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

|                    |                           |
|--------------------|---------------------------|
| <code>__s</code>   | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |
| <code>__f</code>   | Source of the locale.     |
|                    |                           |

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

**Returns**

Iterator to first char not parsed.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base< \\_CharT, \\_InIter >::do\\_get\\_date\(\)](#).  
Referenced by [get\(\)](#), and [get\(\)](#).

**do\_get\_date()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_date\(\)](#) for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond date string.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).  
Referenced by [get\\_date\(\)](#).

**do\_get\_monthname()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
 iter_type __beg,
 iter_type __end,
```

```
ios_base & __io,
ios_base::iostate & __err,
tm * __tm) const [protected], [virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_monthname()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond month name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_monthname\(\)](#).

#### do\_get\_time()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_time()` for details.

#### Parameters

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |
| <code>__io</code>  | Source of the locale.     |

|                    |                                  |
|--------------------|----------------------------------|
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_time\(\)](#).

**do\_get\_weekday()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

[get\\_weekday\(\)](#) for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_weekday\(\)](#).

**do\_get\_year()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
```

```
ios_base::iostate & __err,
tm * __tm) const [protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_year()` for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond year.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::__ctype_abstract_base<_CharT, __ctype_tombis_base>` and `std::use_facet()`.

Referenced by `get_year()`.

#### get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier = 0) const [inline]
```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

See also

`do_get()` and `get()`.

#### Parameters

|                    |                           |
|--------------------|---------------------------|
| <code>__s</code>   | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |



|                         |                                  |
|-------------------------|----------------------------------|
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

**Returns**

Iterator to first char not parsed.

References [do\\_get\(\)](#).

**get()** [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 const char_type * __fmt,
 const char_type * __fmtend) const [inline]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strptime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

**Parameters**

|                       |                                  |
|-----------------------|----------------------------------|
| <code>__s</code>      | Start of string to parse.        |
| <code>__end</code>    | End of string to parse.          |
| <code>__io</code>     | Source of the locale.            |
| <code>__err</code>    | Error flags to set.              |
| <code>__tm</code>     | Pointer to struct tm to fill in. |
| <code>__fmt</code>    | Start of the format string.      |
| <code>__fmtend</code> | End of the format string.        |

**Returns**

Iterator to first char not parsed.

References [std::ios\\_base::M\\_getloc\(\)](#), [do\\_get\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::is\(\)](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::narrow\(\)](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::toupper\(\)](#), and [std::use\\_facet\(\)](#).

**get\_date()**

```
template<typename _CharT, typename _InIter>
```

```

iter_type std::time_get< _CharT, _InIter >::get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]

```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

#### Returns

Iterator to first char beyond date string.

References `do_get_date()`.

#### get\_monthname()

```

template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]

```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |
| <code>__io</code>  | Source of the locale.     |

|                    |                                  |
|--------------------|----------------------------------|
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

References [do\\_get\\_monthname\(\)](#).

**get\_time()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

References [do\\_get\\_time\(\)](#).

**get\_weekday()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond weekday name.

References [do\\_get\\_weekday\(\)](#).

#### get\_year()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond year.

References [do\\_get\\_year\(\)](#).

### 6.958.5 Member Data Documentation

#### id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get<_CharT, _InIter>::id [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 6.959 std::time\_get\_byname<\_CharT, \_InIter> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_get\_byname<\_CharT, \_InIter>:



### Public Types

- typedef `_CharT` **char\_type**
- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }
- typedef `_InIter` **iter\_type**

### Public Member Functions

- **time\_get\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_get\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- iter\_type [get](#) (iter\_type \_\_s, iter\_type \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, char \_\_format, char \_\_modifier=0) const

- iter\_type `get` (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const char\_type \* \_\_fmt, const char\_type \* \_\_fmtend) const
- iter\_type `get_date` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- iter\_type `get_monthname` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- iter\_type `get_time` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- iter\_type `get_weekday` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- iter\_type `get_year` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

### Static Public Attributes

- static locale::id id

### Protected Member Functions

- iter\_type `M_extract_name` (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- iter\_type `M_extract_num` (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, int \_\_min, int \_\_max, size\_t \_\_len, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- iter\_type `M_extract_via_format` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const \_CharT \* \_\_format) const
- iter\_type `M_extract_via_format` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const \_CharT \* \_\_format, \_\_time\_get\_state & \_\_state) const
- iter\_type `M_extract_wday_or_month` (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- virtual dateorder `do_date_order` () const
- iter\_type `do_get` (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_f, ios\_base::iostate & \_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier) const
- virtual iter\_type `do_get_date` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual iter\_type `do_get_monthname` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual iter\_type `do_get_time` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual iter\_type `do_get_weekday` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual iter\_type `do_get_year` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

### Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### 6.959.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >
```

class time\_get\_byname [22.2.5.2].

## 6.959.2 Member Function Documentation

### date\_order()

```
template<typename _CharT, typename _InIter>
```

```
dateorder std::time_get<_CharT, _InIter>::date_order () const [inline], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_←put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `time_base::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

#### Returns

A member of `time_base::dateorder`.

References [do\\_date\\_order\(\)](#).

### do\_date\_order()

```
template<typename _CharT, typename _InIter>
```

```
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get<_CharT, _InIter>::do_←date_order () const [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_←put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of `time_base::dateorder`.

Referenced by [date\\_order\(\)](#).

### do\_get()

```
template<typename _CharT, typename _InIter>
```

```
_InIter std::time_get<_CharT, _InIter>::do_get (
 iter_type __s,
 iter_type __end,
 ios_base & __f,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier) const [inline], [protected], [inherited]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

[get\(\)](#) for more details.

#### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__s</code> | Start of string to parse. |
|------------------|---------------------------|

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__end</code>      | End of string to parse.          |
| <code>__f</code>        | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

#### Returns

Iterator to first char not parsed.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::goodbit](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_](#)  
Referenced by [get\(\)](#), and [get\(\)](#).

#### do\_get\_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_date\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond date string.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).  
Referenced by [get\\_date\(\)](#).



**do\_get\_monthname()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_monthname() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_monthname\(\)](#).

**do\_get\_time()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get\\_time\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond time string.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_time\(\)](#).

#### do\_get\_weekday()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get\\_weekday\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond weekday name.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), and [std::use\\_facet\(\)](#).

Referenced by [get\\_weekday\(\)](#).

**do\_get\_year()**

```
template<typename _CharT, typename _InIter>
_InIter std::time_get<_CharT, _InIter>::do_get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get\\_year\(\)](#) for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT, \\_InIter>](#) and [std::use\\_facet\(\)](#).

Referenced by [get\\_year\(\)](#).

**get()** [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier = 0) const [inline], [inherited]
```

Parse input string according to format.

This function calls [time\\_get::do\\_get](#) with the provided parameters.

See also

`do_get()` and `get()`.

#### Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

#### Returns

Iterator to first char not parsed.

References [do\\_get\(\)](#).

#### `get()` [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 const char_type * __fmt,
 const char_type * __fmtend) const [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strptime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

#### Parameters

|                       |                                  |
|-----------------------|----------------------------------|
| <code>__s</code>      | Start of string to parse.        |
| <code>__end</code>    | End of string to parse.          |
| <code>__io</code>     | Source of the locale.            |
| <code>__err</code>    | Error flags to set.              |
| <code>__tm</code>     | Pointer to struct tm to fill in. |
| <code>__fmt</code>    | Start of the format string.      |
| <code>__fmtend</code> | End of the format string.        |

**Returns**

Iterator to first char not parsed.

References [std::ios\\_base::M\\_getloc\(\)](#), [do\\_get\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), [std::ios\\_base::goodbit](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::is\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::narrow\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::toupper\(\)](#), and [std::use\\_facet\(\)](#).

**get\_date()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

**Returns**

Iterator to first char beyond date string.

References [do\\_get\\_date\(\)](#).

**get\_monthname()**

```
template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <code>__beg</code> | Start of string to parse.                     |
| <code>__end</code> | End of string to parse.                       |
| <code>__io</code>  | Source of the locale.                         |
| <code>__err</code> | Error flags to set.                           |
| <code>__tm</code>  | Pointer to struct <code>tm</code> to fill in. |

#### Returns

Iterator to first char beyond month name.

References [do\\_get\\_monthname\(\)](#).

#### get\_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input time string.

This function parses a time according to the format `X` and puts the results into a user-supplied struct `tm`. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format `X`, `tm` will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                               |
|--------------------|-----------------------------------------------|
| <code>__beg</code> | Start of string to parse.                     |
| <code>__end</code> | End of string to parse.                       |
| <code>__io</code>  | Source of the locale.                         |
| <code>__err</code> | Error flags to set.                           |
| <code>__tm</code>  | Pointer to struct <code>tm</code> to fill in. |

#### Returns

Iterator to first char beyond time string.

References [do\\_get\\_time\(\)](#).

#### get\_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
```

```

iter_type __beg,
iter_type __end,
ios_base & __io,
ios_base::iostate & __err,
tm * __tm) const [inline], [inherited]

```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

### Returns

Iterator to first char beyond weekday name.

References [do\\_get\\_weekday\(\)](#).

### get\_year()

```

template<typename _CharT, typename _InIter>
iter_type std::time_get<_CharT, _InIter>::get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]

```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

### Parameters

|                    |                           |
|--------------------|---------------------------|
| <code>__beg</code> | Start of string to parse. |
| <code>__end</code> | End of string to parse.   |
| <code>__io</code>  | Source of the locale.     |
| <code>__err</code> | Error flags to set.       |

|                   |                                  |
|-------------------|----------------------------------|
| <code>__tm</code> | Pointer to struct tm to fill in. |
|-------------------|----------------------------------|

#### Returns

Iterator to first char beyond year.

References [do\\_get\\_year\(\)](#).

### 6.959.3 Member Data Documentation

#### id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

### 6.960 `std::chrono::time_point< _Clock, _Dur >` Class Template Reference

#### Public Types

- typedef `_Clock` **clock**
- typedef `_Dur` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

#### Public Member Functions

- constexpr **time\_point** (const duration &\_\_dur)
- template<typename \_Dur2, typename = \_Require<is\_convertible<\_Dur2, \_Dur>>>> constexpr **time\_point** (const time\_point< clock, \_Dur2 > &\_\_t)
- constexpr time\_point & **operator++** ()
- constexpr time\_point **operator++** (int)
- constexpr time\_point & **operator+=** (const duration &\_\_dur)
- constexpr time\_point & **operator--** ()
- constexpr time\_point **operator--** (int)
- constexpr time\_point & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

#### Static Public Member Functions

- static constexpr time\_point **max** () noexcept
- static constexpr time\_point **min** () noexcept

#### Related Symbols

(Note that these are not member symbols.)

- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2> constexpr time\_point< \_Clock, typename [common\\_type](#)< \_Dur1, duration< \_Rep2, \_Period2 > >::type > [operator+](#) (const time\_point< \_Clock, \_Dur1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)



### 6.960.1 Detailed Description

```
template<typename _Clock, typename _Dur>
class std::chrono::time_point< _Clock, _Dur >
```

`chrono::time_point` represents a point in time as measured by a clock  
 The documentation for this class was generated from the following file:

- [chrono.h](#)

## 6.961 std::time\_put<\_CharT, \_OutIter> Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for `std::time_put<_CharT, _OutIter>`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_OutIter` [iter\\_type](#)

### Public Member Functions

- [time\\_put](#) (`size_t __refs=0`)
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const `tm * __tm`, char \_\_format, char \_\_mod=0) const
- [iter\\_type put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const `tm * __tm`, const `_CharT * __beg`, const `_CharT * __end`) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~time\\_put](#) ()
- virtual [iter\\_type do\\_put](#) ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

### Static Protected Member Functions

- static [\\_\\_c\\_locale \\_S\\_clone\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc, const char \*\_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) ([\\_\\_c\\_locale](#) &\_\_cloc)
- static [\\_\\_c\\_locale \\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static [\\_\\_c\\_locale \\_S\\_lc\\_ctype\\_c\\_locale](#) ([\\_\\_c\\_locale](#) \_\_cloc, const char \*\_\_s)

#### 6.961.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::time_put< _CharT, _OutIter >
```

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

The `time_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_put` facet.

#### 6.961.2 Member Typedef Documentation

##### `char_type`

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::time_put< _CharT, _OutIter >::char_type
```

Public typedefs.

##### `iter_type`

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

#### 6.961.3 Constructor & Destructor Documentation

##### `time_put()`

```
template<typename _CharT, typename _OutIter>
std::time_put< _CharT, _OutIter >::time_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

References [std::locale::facet::facet\(\)](#).

**~time\_put()**

```
template<typename _CharT, typename _OutIter>
virtual std::time_put< _CharT, _OutIter >::~~time_put () [inline], [protected], [virtual]
Destructor.
```

**6.961.4 Member Function Documentation****do\_put()**

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod) const [protected], [virtual]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

**See also**

[put\(\)](#) for more details.

**Parameters**

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

References [std::ios\\_base::M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::widen\(\)](#).

Referenced by [put\(\)](#), and [put\(\)](#).

**put() [1/2]**

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
```

```
char __format,
char __mod = 0) const [inline]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

#### Returns

Iterator after writing.

References [do\\_put\(\)](#).

#### put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 const _CharT * __beg,
 const _CharT * __end) const
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

#### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The stream to write to.                      |
| <code>__io</code>   | Source of locale.                            |
| <code>__fill</code> | char_type to use for padding.                |
| <code>__tm</code>   | Struct tm with date and time info to format. |
| <code>__beg</code>  | Start of format string.                      |
| <code>__end</code>  | End of format string.                        |

**Returns**

Iterator after writing.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [do\\_put\(\)](#), [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::narrow\(\)](#), and [std::use\\_facet\(\)](#).

**6.961.5 Member Data Documentation****id**

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id [static]
Numpunct facet id.
```

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

**6.962 std::time\_put\_byname< \_CharT, \_OutIter > Class Template Reference**

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time\_put\_byname< \_CharT, \_OutIter >:

**Public Types**

- typedef `_CharT` **char\_type**
- typedef `_OutIter` **iter\_type**

**Public Member Functions**

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_put\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- iter\_type **put** (iter\_type \_\_s, [ios\\_base](#) &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

- iter\_type `put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, const \_CharT \* \_\_beg, const \_CharT \* \_\_end) const

### Static Public Attributes

- static `locale::id` id

### Protected Member Functions

- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, const tm \* \_\_tm, char \_\_format, char \_\_mod) const

### Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## 6.962.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::time_put_byname< _CharT, _OutIter >
```

class time\_put\_byname [22.2.5.4].

## 6.962.2 Member Function Documentation

### do\_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod) const [protected], [virtual], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

### See also

`put()` for more details.

### Parameters

|                   |                         |
|-------------------|-------------------------|
| <code>__s</code>  | The stream to write to. |
| <code>__io</code> | Source of locale.       |

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::use\\_facet\(\)](#), and [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >::widen\(\)](#).

Referenced by [put\(\)](#), and [put\(\)](#).

**put()** [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod = 0) const [inline], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

**Parameters**

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

References [do\\_put\(\)](#).

**put()** [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
```

```

ios_base & __io,
char_type __fill,
const tm * __tm,
const _CharT * __beg,
const _CharT * __end) const [inherited]

```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

#### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The stream to write to.                      |
| <code>__io</code>   | Source of locale.                            |
| <code>__fill</code> | char_type to use for padding.                |
| <code>__tm</code>   | Struct tm with date and time info to format. |
| <code>__beg</code>  | Start of format string.                      |
| <code>__end</code>  | End of format string.                        |

#### Returns

Iterator after writing.

References [std::ios\\_base::M\\_getloc\(\)](#), [do\\_put\(\)](#), [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>::narrow\(\)](#), and [std::use\\_facet\(\)](#).

### 6.962.3 Member Data Documentation

#### id

```

template<typename _CharT, typename _OutIter>
locale::id std::time_put<_CharT, _OutIter>::id [static], [inherited]

```

Numpunct facet id.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 6.963 std::timed\_mutex Class Reference

```
#include <mutex>
```

#### Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

#### Public Member Functions

- **timed\_mutex** (const timed\_mutex &)=delete
- void **lock** ()
- native\_handle\_type **native\_handle** () noexcept
- timed\_mutex & **operator=** (const timed\_mutex &)=delete
- bool **try\_lock** () noexcept



- template<class \_Rep, class \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<class \_Clock, class \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

**Friends**

- class **\_\_timed\_mutex\_impl**< **timed\_mutex** >

**6.963.1 Detailed Description**

The standard timed mutex type.

A non-recursive mutex that supports a timeout when trying to acquire the lock.

Since

C++11

The documentation for this class was generated from the following file:

- [mutex](#)

**6.964 std::to\_chars\_result Struct Reference**

```
#include <charconv>
```

**Public Member Functions**

- constexpr **operator bool** () const noexcept

**Public Attributes**

- errc **ec**
- char \* **ptr**

**Friends**

- bool **operator==** (const [to\\_chars\\_result](#) &, const [to\\_chars\\_result](#) &)=default

**6.964.1 Detailed Description**

Result type of std::to\_chars.

The documentation for this struct was generated from the following file:

- [charconv](#)

**6.965 std::chrono::treat\_as\_floating\_point< \_Rep > Struct Template Reference**

```
#include <chrono.h>
```

Inheritance diagram for `std::chrono::treat_as_floating_point<_Rep>`:



#### 6.965.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::treat_as_floating_point<_Rep>
```

Trait indicating whether to treat a type as a floating-point type.

The chrono library uses this trait to tell whether a `duration` can represent fractional values of the given precision, or only integral values.

You should specialize this trait for your own numeric types that are used with `duration` and can represent non-integral values.

Since

C++11

The documentation for this struct was generated from the following file:

- [chrono.h](#)

#### 6.966 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef Cmp\_Fn `cmp_fn`
- typedef Node\_Update `node_update`

## Public Member Functions

- `tree` (const `cmp_fn` &c)
- `tree` (const tree &other)
- template<typename It>  
`tree` (It first, It last)
- template<typename It>  
`tree` (It first, It last, const `cmp_fn` &c)
- tree & `operator=` (const tree &other)
- void `swap` (tree &other)

### 6.966.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_
tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class
Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >

```

A tree-based container.

### Template Parameters

|                    |                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                             |
| <i>Mapped</i>      | Map type.                                                                                                                             |
| <i>Cmp_Fn</i>      | Comparison functor.                                                                                                                   |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> .                                                                   |
| <i>Node_Update</i> | Updates tree internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                       |

Base tag choices are: `ov_tree_tag`, `rb_tree_tag`, `splay_tree_tag`.

Base is `basic_branch`.

### 6.966.2 Member Typedef Documentation

#### `cmp_fn`

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
Comparison functor type.
```

### 6.966.3 Constructor & Destructor Documentation

#### `tree()` [1/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 const cmp_fn & c) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

#### `tree()` [2/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

#### `tree()` [3/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↵
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 It first,
```

```

 It last,
 const cmp_fn & c) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object. The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 6.967 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

### 6.967.1 Detailed Description

```

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >

```

Tree metadata helper.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.968 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

```

#include <node_metadata_selector.hpp>

```

### Public Types

- typedef `Node_Update::metadata_type` **type**

### 6.968.1 Detailed Description

```

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

```

Specialization, false.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.969 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >` Struct Template Reference

```

#include <node_metadata_selector.hpp>

```

### Public Types

- typedef `null_type` **type**

### 6.969.1 Detailed Description

```

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

```

Specialization, true.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.970 `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

### Public Types

- typedef `tree_metadata_helper< __node_u, null_update >::type` **type**

#### 6.970.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename
Const_Itrator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.971 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >` Class Template Reference

```
#include <tree_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**

- typedef size\_type **metadata\_type**
- typedef Node\_Cltr **node\_const\_iterator**
- typedef Node\_Itr **node\_iterator**
- typedef allocator\_type::size\_type **size\_type**

### Public Member Functions

- iterator [find\\_by\\_order](#) (size\_type)
- const\_iterator [find\\_by\\_order](#) (size\_type) const
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const

### Protected Member Functions

- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

#### 6.971.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>`  
`class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`

Functor updating ranks of entrees.

#### 6.971.2 Member Function Documentation

##### [find\\_by\\_order\(\)](#) [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statist
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
 size_type order) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

##### [find\\_by\\_order\(\)](#) [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_s
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
 size_type order) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

##### [operator\(\)](#)

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
void __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::↵
operator() (
 node_iterator node_it,
 node_const_iterator end_nd_it) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

##### [order\\_of\\_key\(\)](#)

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statist
```

```
Node_CIter, Node_Iter, Cmp_Fn, _Alloc >::order_of_key (
 key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

The documentation for this class was generated from the following file:

- [tree\\_policy.hpp](#)

## 6.972 \_\_gnu\_pbds::tree\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::tree_tag`:



### 6.972.1 Detailed Description

Basic tree structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.973 \_\_gnu\_pbds::detail::tree\_traits< Key, Data, Cmp\_Fn, Node\_Update, Tag, \_Alloc > Struct Template Reference

### 6.973.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CIter, typename
Node_Iter, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >
```

Tree traits class, primary template.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)



## 6.974 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct` Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` `metadata_type`
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` `node_const_iterator`
- typedef `ov_tree_node_it_< value_type, metadata_type, _Alloc >` `node_iterator`
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` `node_update`
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_`  
`_update_pointer`

#### 6.974.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr,
class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

#### 6.974.2 Member Typedef Documentation

##### `node_const_iterator`

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr,
class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

## 6.975 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct` Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc`

&gt;:



## Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef [rb\\_tree\\_node](#) < [types\\_traits](#)< Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#)< Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc > **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [rb\\_tree\\_node](#) < [types\\_traits](#)< Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#)< Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, point\_const\_iterator, point\_iterator, \_Alloc > [node\\_const\\_iterator](#)
- typedef [bin\\_search\\_tree\\_node\\_it](#) < [rb\\_tree\\_node](#) < [types\\_traits](#)< Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#)< Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, point\_const\_iterator, point\_iterator, \_Alloc > **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), node\_iterator, Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#)< [node\\_const\\_iterator](#), node\_iterator, Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename type\_traits::value\_type, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **reverse\_iterator**

### 6.975.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

## 6.975.2 Member Typedef Documentation

### node\_const\_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type,
tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc
>::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

## 6.976 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**

- typedef `__gnu_pbds::null_node_update`< `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > \* `null_node_update_pointer`
- typedef `bin_search_tree_const_it`< `typename node_alloc_traits::pointer`, `typename type_traits::value_type`, `typename type_traits::pointer`, `typename type_traits::const_pointer`, `typename type_traits::reference`, `typename type_traits::const_reference`, `true`, `_Alloc` > `point_const_iterator`
- typedef `bin_search_tree_it`< `typename node_alloc_traits::pointer`, `typename type_traits::value_type`, `typename type_traits::pointer`, `typename type_traits::const_pointer`, `typename type_traits::reference`, `typename type_traits::const_reference`, `true`, `_Alloc` > `point_iterator`
- typedef `bin_search_tree_it`< `typename node_alloc_traits::pointer`, `typename type_traits::value_type`, `typename type_traits::pointer`, `typename type_traits::const_pointer`, `typename type_traits::reference`, `typename type_traits::const_reference`, `false`, `_Alloc` > `reverse_iterator`

### 6.976.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

### 6.976.2 Member Typedef Documentation

#### `node_const_iterator`

```
typedef bin_search_tree_const_node_it< splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

### 6.977 `__gnu_pbds::detail::tree_traits`< `Key`, `null_type`, `Cmp_Fn`, `Node_Update`, `ov_tree_tag`, `_Alloc` > Struct Template Reference

```
#include <traits.hpp>
```

#### Public Types

- typedef `tree_node_metadata_dispatch`< `Key`, `null_type`, `Cmp_Fn`, `Node_Update`, `_Alloc` >::type `metadata_type`
- typedef `ov_tree_node_const_it`< `value_type`, `metadata_type`, `_Alloc` > `node_const_iterator`
- typedef `node_const_iterator` `node_iterator`
- typedef `Node_Update`< `node_const_iterator`, `node_const_iterator`, `Cmp_Fn`, `_Alloc` > `node_update`
- typedef `__gnu_pbds::null_node_update`< `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > \* `null_node_update_pointer`

### 6.977.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Specialization.

## 6.977.2 Member Typedef Documentation

### node\_const\_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn↵
_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

## 6.978 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**

- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 6.978.1 Detailed Description

```
template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

### 6.978.2 Member Typedef Documentation

#### `node_const_iterator`

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

### 6.979 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

### 6.979.1 Detailed Description

`template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_↵, typename _Alloc_ > class Node_Update, typename _Alloc>`  
`struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`

Specialization.

### 6.979.2 Member Typedef Documentation

#### `node_const_iterator`

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc
>::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc
>::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

The documentation for this struct was generated from the following file:

- `splay_tree_/traits.hpp`

## 6.980 `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >` Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef `_ATraits` [access\\_traits](#)
- typedef `Node_Update` **node\_update**

## Public Member Functions

- [trie](#) (const [access\\_traits](#) &t)
- **trie** (const trie &other)
- template<typename It>  
[trie](#) (It first, It last)
- template<typename It>  
[trie](#) (It first, It last, const [access\\_traits](#) &t)
- trie & **operator=** (const trie &other)
- void **swap** (trie &other)

### 6.980.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
```

```
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >
```

A trie-based container.



**Template Parameters**

|                    |                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                             |
| <i>Mapped</i>      | Map type.                                                                                                                             |
| <i>_ATraits</i>    | Element access traits.                                                                                                                |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> .                                                                   |
| <i>Node_Update</i> | Updates trie internal-nodes, restores invariants when invalidated. XXX See <code>design::tree-based-containersnode</code> invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                       |

Base tag choice is `pat_trie_tag`.

Base is `basic_branch`.

**6.980.2 Member Typedef Documentation****access\_traits**

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
 typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr, typename _ATraits_,
 typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
```

```
typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits
```

Element access traits type.

**6.980.3 Constructor & Destructor Documentation****trie() [1/3]**

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
 typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr, typename _ATraits_,
 typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
```

```
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 const access_traits & t) [inline]
```

Constructor taking some policy objects. `r_access_traits` will be copied by the `_ATraits` object of the container object.

**trie() [2/3]**

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
 typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr, typename _ATraits_,
 typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
```

```
template<typename It>
```

```
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

**trie() [3/3]**

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type,
 typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_CItr, typename _ATraits_,
 typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
```

```
itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 It first,
 It last,
 const access_traits & t) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects. The value\_types between first\_it and last\_it will be inserted into the container object.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 6.981 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, \_BTp > Struct Template Reference

### 6.981.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.982 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

### Public Types

- typedef Node\_Update::metadata\_type type

### 6.982.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

## 6.983 \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, true > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

### Public Types

- typedef null\_type type

#### 6.983.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, true.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 6.984 **\_\_gnu\_pbds::detail::trie\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc > Struct Template Reference**

```
#include <node_metadata_selector.hpp>
```

#### Public Types

- typedef [trie\\_metadata\\_helper](#)< \_\_node\_u, null\_update >::type **type**

#### 6.984.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename
Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 6.985 **\_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference**

```
#include <trie_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Citr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `access_traits::const_iterator` **a\_const\_iterator**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Citr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Public Member Functions

- iterator [find\\_by\\_order](#) (size\_type)
- const\_iterator [find\\_by\\_order](#) (size\_type) const
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const
- size\_type [order\\_of\\_prefix](#) (a\_const\_iterator, a\_const\_iterator) const

## Protected Member Functions

- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

## Private Member Functions

- virtual const\_iterator [end](#) () const =0
- virtual const access\_traits & [get\\_access\\_traits](#) () const =0

### 6.985.1 Detailed Description

template<typename Node\_CIttr, typename Node\_Itr, typename \_ATraits, typename \_Alloc>  
class \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_CIttr, Node\_Itr, \_ATraits, \_Alloc >

Functor updating ranks of entrees.

### 6.985.2 Member Function Documentation

#### [find\\_by\\_order\(\)](#) [1/2]

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_stat
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
 size_type order) [inline]
```

Finds an entry by \_\_order. Returns an iterator to the entry with the \_\_order order, or an iterator to the container object's end if order is at least the size of the container object.

#### [find\\_by\\_order\(\)](#) [2/2]

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
 size_type order) const [inline]
```

Finds an entry by \_\_order. Returns a const\_iterator to the entry with the \_\_order order, or a const\_iterator to the container object's end if order is at least the size of the container object.

#### [operator>\(\)](#)

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::↵
operator() (
 node_iterator nd_it,
 node_const_iterator) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator node\_it; end\_nd\_it is the end node iterator.

#### [order\\_of\\_key\(\)](#)

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_stat
Node_CIttr, Node_Itr, _ATraits, _Alloc >::order_of_key (
 key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if r\_key is the smallest key, this method will return 0; if r\_key is a key between the smallest and next key, this method will return 1; if r\_key is a key larger than the largest key, this method will return the size of r\_c.

## order\_of\_prefix()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_statistics_node_update<
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
 a_const_iterator b,
 a_const_iterator e) const [inline]
```

Returns the order of a prefix within a sequence. For example, if [b, e) is the smallest prefix, this method will return 0; if r\_key is a key between the smallest and next key, this method will return 1; if r\_key is a key larger than the largest key, this method will return the size of r\_c.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

## 6.986 \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference

```
#include <trie_policy_base.hpp>
```

Inheritance diagram for \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc >:



## Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_CItr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**

- typedef [rebind\\_traits](#)< \_Alloc, rkey\_type > **rebind\_k**
- typedef [rebind\\_traits](#)< \_Alloc, rvalue\_type > **rebind\_v**
- typedef [rebind\\_v](#)::reference **reference**
- typedef [std::iterator\\_traits](#)< it\_type >::value\_type **value\_type**

### Protected Member Functions

- virtual const\_iterator **end** () const =0
- virtual iterator [end](#) ()=0
- it\_type **end\_iterator** () const
- virtual const access\_traits & **get\_access\_traits** () const =0
- virtual node\_const\_iterator **node\_begin** () const =0
- virtual node\_iterator **node\_begin** ()=0
- virtual node\_const\_iterator **node\_end** () const =0
- virtual node\_iterator **node\_end** ()=0

### Static Protected Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

#### 6.986.1 Detailed Description

`template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>`  
**class** `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`

Base class for trie policies.

#### 6.986.2 Member Function Documentation

##### **end()**

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
virtual iterator __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc
>::end () [protected], [pure virtual]
```

Implements [\\_\\_gnu\\_pbds::detail::branch\\_policy](#)< Node\_Cltr, Node\_Itr, \_Alloc >.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

#### 6.987 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

```
#include <trie_policy.hpp>
```

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `access_traits::const_iterator` [a\\_const\\_iterator](#)
- typedef `_ATraits` [access\\_traits](#)
- typedef `_Alloc` [allocator\\_type](#)
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef [null\\_type](#) **metadata\_type**
- typedef `Node_CItr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` [size\\_type](#)

## Public Member Functions

- `std::pair< iterator, iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#))
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) ([a\\_const\\_iterator](#), [a\\_const\\_iterator](#)) const
- `std::pair< iterator, iterator >` [prefix\\_range](#) (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` [prefix\\_range](#) (`key_const_reference`) const



## Protected Member Functions

- void `operator()` (node\_iterator node\_it, node\_const\_iterator end\_nd\_it) const

### 6.987.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A node updatator that allows tries to be searched for the range of values that match a certain prefix.

### 6.987.2 Member Typedef Documentation

#### `a_const_iterator`

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_Cltr,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
Const element iterator.
```

#### `access_traits`

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _↵
Alloc >::access_traits
Element access traits.
```

#### `allocator_type`

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc
>::allocator_type
_Alloc type.
```

#### `size_type`

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_↵
Itr, _ATraits, _Alloc >::size_type
Size type.
```

### 6.987.3 Member Function Documentation

#### `operator>()`

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::↵
operator() (
 node_iterator node_it,
 node_const_iterator end_nd_it) const [inline], [protected]
```

Called to update a node's metadata.

#### `prefix_range()` [1/4]

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::↵
iterator, typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::↵
```

```

iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
 a_const_iterator b,
 a_const_iterator e)

```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

#### prefix\_range() [2/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↵
const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
 a_const_iterator b,
 a_const_iterator e) const

```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

#### prefix\_range() [3/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↵
iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↵
iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
 key_const_reference r_key)

```

Finds the iterator range corresponding to all values whose prefixes match r\_key.

#### prefix\_range() [4/4]

```

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↵
const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
 key_const_reference r_key) const

```

Finds the const iterator range corresponding to all values whose prefixes match r\_key.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

### 6.988 \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc > Struct Template Reference

```
#include <trie_policy.hpp>
```

#### Public Types

- enum { **reverse** }
- enum { **min\_e\_val** , **max\_e\_val** , **max\_size** }
- typedef detail::\_\_conditional\_type< Reverse, typenameString::const\_reverse\_iterator, typenameString::const\_↵  
iterator >::\_\_type **const\_iterator**
- typedef std::iterator\_traits< const\_iterator >::value\_type **e\_type**
- typedef detail::rebind\_traits< \_Alloc, key\_type >::const\_reference **key\_const\_reference**
- typedef String **key\_type**
- typedef \_Alloc::size\_type **size\_type**

## Static Public Member Functions

- static [const\\_iterator begin](#) (key\_const\_reference)
- static size\_type [e\\_pos](#) ([e\\_type](#) e)
- static [const\\_iterator end](#) (key\_const\_reference)

### 6.988.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_↵
traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_↵
traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>
struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

## Template Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>String</i>    | String type.                                      |
| <i>Min_E_Val</i> | Minimal element value.                            |
| <i>Max_E_Val</i> | Maximum element value.                            |
| <i>Reverse</i>   | Reverse iteration should be used. Default: false. |
| <i>_Alloc</i>    | Allocator type.                                   |

### 6.988.2 Member Typedef Documentation

#### const\_iterator

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__↵
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__↵
__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =
std::allocator<char>>
typedef detail::__conditional_type<Reverse,typenameString::const_reverse_iterator,typename↵
String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_↵
E_Val, Reverse, _Alloc >::__const_iterator
```

Element const iterator type.

#### e\_type

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__↵
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__↵
__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =
std::allocator<char>>
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

### 6.988.3 Member Function Documentation

#### begin()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_↵
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::__const_iterator __gnu_pbds::trie_strin
```

```
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (
 key_const_reference r_key) [inline], [static]
```

Returns a `const_iterator` to the first element of `key_const_reference` argument.

### **e\_pos()**

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max←
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_ac
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
 e_type e) [inline], [static]
```

Maps an element to a position.

### **end()**

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max←
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_strin
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
 key_const_reference r_key) [inline], [static]
```

Returns a `const_iterator` to the after-last element of `key_const_reference` argument.

The documentation for this struct was generated from the following file:

- [trie\\_policy.hpp](#)

## **6.989 \_\_gnu\_pbds::trie\_tag Struct Reference**

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for `__gnu_pbds::trie_tag`:



#### 6.989.1 Detailed Description

Basic trie structure.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 6.990 `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >` Struct Template Reference

##### 6.990.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
```

Trie traits class, primary template.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 6.991 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::Head< synth_access_traits, metadata >` **head**
- typedef `base_type::Inode< synth_access_traits, metadata >` **inode**
- typedef `base_type::Iter< node, leaf, head, inode, true >` **iterator**
- typedef `base_type::Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `base_type::Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > * null_node_update_pointer`
- typedef `base_type::Iter< node, leaf, head, inode, false >` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits >` **synth\_access\_traits**

### 6.991.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

### 6.991.2 Member Typedef Documentation

#### **node\_const\_iterator**

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
```

```
typedef base_type::Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tr
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

#### **node\_update**

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
```

```
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

## synth\_access\_traits

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_traits
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 6.992 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

```
#include <traits.hpp>
```

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::CIter< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::CIter< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::Head< synth\_access\_traits, metadata >` **head**
- typedef `base_type::Inode< synth\_access\_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::Leaf< synth\_access\_traits, metadata >` **leaf**
- typedef `base_type::Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null\_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::Node_base< synth\_access\_traits, metadata >` **node**
- typedef `base_type::Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node\_const\_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node\_const\_iterator, node\_const\_iterator, _ATraits, _Alloc > * null\_node\_update` **←\_pointer**
- typedef `const_reverse_iterator` **reverse\_iterator**
- typedef `\_\_gnu\_pbds::detail::synth\_access\_traits< type_traits, true, access_traits >` **[synth\\_access\\_traits](#)**

### 6.992.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

### 6.992.2 Member Typedef Documentation

#### node\_const\_iterator

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

### node\_update

```
template<typename Key, typename _ATraits, template< typename Node_CIter, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
Type for node update.
```

### synth\_access\_traits

```
template<typename Key, typename _ATraits, template< typename Node_CIter, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
Type for synthesized traits.
```

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 6.993 \_\_gnu\_pbds::trivial\_iterator\_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

### 6.993.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 6.994 std::try\_to\_lock\_t Struct Reference

```
#include <std_mutex.h>
```

### 6.994.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 6.995 std::tuple<\_Elements> Class Template Reference

```
#include <tuple>
```

### Public Member Functions

- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor<_Valid, _UElements...> = false> constexpr tuple (_UElements &&... __elements) noexcept(__nothrow_constructible<_UElements...>())`
- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor<_Valid, _UElements...> = true> constexpr tuple (_UElements &&... __elements) noexcept(__nothrow_constructible<_UElements...>())`
- `template<typename _Alloc, _ExplicitDefaultCtor<is_object<_Alloc>::value> = false> constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc, _ImplicitDefaultCtor<is_object<_Alloc>::value> = true> constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`



- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor<_Valid, _UElements...> = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor<_Valid, _UElements...> = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `template<typename _Alloc, bool _NotEmpty = (sizeof...( _Elements) >= 1), _ExplicitCtor<_NotEmpty, const _Elements &... > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const __type_identity_t< _Elements > &... __elements)`
- `template<typename _Alloc, bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor<_NotEmpty, const _Elements &... > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const __type_identity_t< _Elements > &... __elements)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_`  
`ctor<const tuple<_UElements...>&>(), _ExplicitCtor<_Valid, const _UElements &... > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_`  
`ctor<const tuple<_UElements...>&>(), _ImplicitCtor<_Valid, const _UElements &... > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_`  
`ctor<tuple<_UElements...>&>(), _ExplicitCtor<_Valid, _UElements... > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_`  
`ctor<tuple<_UElements...>&>(), _ImplicitCtor<_Valid, _UElements... > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ExplicitCtor<_NotEmpty, const _Elements &... > = false>`  
`constexpr tuple (const __type_identity_t< _Elements > &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor<_NotEmpty, const _Elements &... > = true>`  
`constexpr tuple (const __type_identity_t< _Elements > &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `constexpr tuple (const tuple &)=default`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<const tuple<_`  
`UElements...>&>(), _ExplicitCtor<_Valid, const _UElements &... > = false>`  
`constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<const tuple<_`  
`UElements...>&>(), _ImplicitCtor<_Valid, const _UElements &... > = true>`  
`constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `constexpr tuple (tuple &&)=default`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<tuple<_`  
`UElements...>&&(), _ExplicitCtor<_Valid, _UElements... > = false>`  
`constexpr tuple (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !_use_other_ctor<tuple<_`  
`UElements...>&&(), _ImplicitCtor<_Valid, _UElements... > = true>`  
`constexpr tuple (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())`
- `constexpr tuple & operator= (__conditional_t< __assignable< _Elements... >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _Elements... >())`

- constexpr tuple & **operator=** (\_\_conditional\_t< \_\_assignable< const \_Elements &... >(), const tuple &, const \_\_nonenum & > \_\_in) noexcept(\_\_nothrow\_assignable< const \_Elements &... >())
- template<typename... \_UElements>  
constexpr \_\_enable\_if\_t< \_\_assignable< const \_UElements &... >(), tuple & > **operator=** (const tuple< \_UElements... > & \_\_in) noexcept(\_\_nothrow\_assignable< const \_UElements &... >())
- template<typename... \_UElements>  
constexpr \_\_enable\_if\_t< \_\_assignable< \_UElements... >(), tuple & > **operator=** (tuple< \_UElements... > & \_\_in) noexcept(\_\_nothrow\_assignable< \_UElements... >())
- constexpr void **swap** (tuple & \_\_in) noexcept(\_\_and< \_\_is\_nothrow\_swappable< \_Elements >... >::value)

### 6.995.1 Detailed Description

template<typename... \_Elements>

class std::tuple< \_Elements >

Primary class template, tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 6.996 std::tuple< \_T1, \_T2 > Class Template Reference

```
#include <tuple>
```

### Public Member Functions

- template<typename \_U1, typename \_U2, \_ExplicitCtor<! \_\_is\_alloc\_arg< \_U1 >(), \_U1, \_U2 > = false>  
constexpr tuple ( \_U1 && \_\_a1, \_U2 && \_\_a2) noexcept(\_\_nothrow\_constructible< \_U1, \_U2 >())
- template<typename \_U1, typename \_U2, \_ImplicitCtor<! \_\_is\_alloc\_arg< \_U1 >(), \_U1, \_U2 > = true>  
constexpr tuple ( \_U1 && \_\_a1, \_U2 && \_\_a2) noexcept(\_\_nothrow\_constructible< \_U1, \_U2 >())
- template<typename \_Alloc, \_ExplicitDefaultCtor< [is\\_object](#)< \_Alloc >::value, \_T1, \_T2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a)
- template<typename \_Alloc, \_ImplicitDefaultCtor< [is\\_object](#)< \_Alloc >::value, \_T1, \_T2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, \_U1 && \_\_a1, \_U2 && \_\_a2)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, \_U1, \_U2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, \_U1 && \_\_a1, \_U2 && \_\_a2)
- template<typename \_Alloc, bool \_Dummy = true, \_ExplicitCtor< \_Dummy, const \_T1 &, const \_T2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const \_T1 & \_\_a1, const \_T2 & \_\_a2)
- template<typename \_Alloc, bool \_Dummy = true, \_ImplicitCtor< \_Dummy, const \_T1 &, const \_T2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const \_T1 & \_\_a1, const \_T2 & \_\_a2)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, const \_U1 &, const \_U2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const [pair](#)< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, const \_U1 &, const \_U2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const [pair](#)< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, const \_U1 &, const \_U2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, const \_U1 &, const \_U2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, [pair](#)< \_U1, \_U2 > && \_\_in)

- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)`
- `template<typename _Alloc>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)`
- `template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)`
- `template<bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `constexpr tuple (const tuple &)=default`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `constexpr tuple (tuple &&)=default`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `constexpr tuple & operator= (__conditional_t< __assignable< _T1, _T2 >(), tuple &&, __nonesuch && > __in)`  
`noexcept(__nothrow_assignable< _T1, _T2 >())`
- `constexpr tuple & operator= (__conditional_t< __assignable< const _T1 &, const _T2 & >(), const tuple &, const __nonesuch & > __in)`  
`noexcept(__nothrow_assignable< const _T1 &, const _T2 & >())`
- `template<typename _U1, typename _U2>`  
`constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > operator= (const pair< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2>`  
`constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > operator= (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2>`  
`constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > operator= (pair< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())`
- `template<typename _U1, typename _U2>`  
`constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > operator= (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())`
- `constexpr void swap (tuple &__in) noexcept(__and< __is_nothrow_swappable< _T1 >, __is_nothrow_swappable< _T2 > >::value)`

### 6.996.1 Detailed Description

```
template<typename _T1, typename _T2>
class std::tuple< _T1, _T2 >
```

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.  
The documentation for this class was generated from the following file:

- [tuple](#)

### 6.997 std::tuple\_element< \_\_i, \_Tp > Struct Template Reference

#### 6.997.1 Detailed Description

```
template<size_t __i, typename _Tp>
struct std::tuple_element< __i, _Tp >
```

Gives the type of the ith element of a given tuple type.  
The documentation for this struct was generated from the following file:

- [utility.h](#)

### 6.998 std::tuple\_element< 0, pair< \_Tp1, \_Tp2 > > Struct Template Reference

```
#include <stl_pair.h>
```

#### Public Types

- typedef \_Tp1 type

#### 6.998.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.  
The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

### 6.999 std::tuple\_element< 1, pair< \_Tp1, \_Tp2 > > Struct Template Reference

```
#include <stl_pair.h>
```

#### Public Types

- typedef \_Tp2 type

#### 6.999.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.  
The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 6.1000 std::tuple\_element< \_\_i, tuple< \_Types... > > Struct Template Reference

```
#include <tuple>
```

### Public Types

- using **type**

#### 6.1000.1 Detailed Description

```
template<size_t __i, typename... _Types>
struct std::tuple_element< __i, tuple< _Types... > >
```

Trait to get the lth element type from a tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 6.1001 std::tuple\_element< \_Ind, array< \_Tp, \_Nm > > Struct Template Reference

```
#include <array>
```

### Public Types

- using **type**

#### 6.1001.1 Detailed Description

```
template<size_t _Ind, typename _Tp, size_t _Nm>
struct std::tuple_element< _Ind, array< _Tp, _Nm > >
```

Partial specialization for std::array.

The documentation for this struct was generated from the following file:

- [array](#)

## 6.1002 std::tuple\_size< \_Tp > Struct Template Reference

Inherited by std::tuple\_size< const \_\_enable\_if\_has\_tuple\_size< \_Tp > >, std::tuple\_size< const volatile \_\_enable\_if\_has\_tuple\_size< \_Tp > >, and std::tuple\_size< volatile \_\_enable\_if\_has\_tuple\_size< \_Tp > >.

#### 6.1002.1 Detailed Description

```
template<typename _Tp>
struct std::tuple_size< _Tp >
```

Finds the size of a given tuple type.

The documentation for this struct was generated from the following file:

- [utility.h](#)

## 6.1003 std::tuple\_size< array< \_Tp, \_Nm > > Struct Template Reference

```
#include <array>
```

Inheritance diagram for `std::tuple_size< array< _Tp, _Nm > >`:



#### Public Types

- using **type**
- using **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept

#### Static Public Attributes

- static constexpr size\_t **value**

#### 6.1003.1 Detailed Description

```
template<typename _Tp, size_t _Nm>
struct std::tuple_size< array< _Tp, _Nm > >
```

Partial specialization for `std::array`.

The documentation for this struct was generated from the following file:

- [array](#)

#### 6.1004 `std::tuple_size< pair< _Tp1, _Tp2 > >` Struct Template Reference

```
#include <stl_pair.h>
```

Inheritance diagram for std::tuple\_size< pair< \_Tp1, \_Tp2 > >:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr size\_t **value**

#### 6.1004.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_size< pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

### 6.1005 std::tuple\_size< tuple< \_Elements... > > Struct Template Reference

```
#include <tuple>
```

Inheritance diagram for `std::tuple_size< tuple< _Elements... > >`:



### Public Types

- using **type**
- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept

### Static Public Attributes

- static constexpr size\_t **value**

#### 6.1005.1 Detailed Description

```
template<typename... _Elements>
struct std::tuple_size< tuple< _Elements... > >
```

class tuple\_size

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 6.1006 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >::type Struct Reference

```
#include <entry_cmp.hpp>
```

### Public Member Functions

- **type** (const Cmp\_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const



### 6.1006.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type
```

Compare plus entry.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 6.1007 std::type\_index Struct Reference

```
#include <typeindex>
```

### Public Member Functions

- **type\_index** (const [type\\_info](#) &\_\_rhs) noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const type\_index &\_\_rhs) const noexcept
- bool **operator<** (const type\_index &\_\_rhs) const noexcept
- bool **operator<=** (const type\_index &\_\_rhs) const noexcept
- bool **operator==** (const type\_index &\_\_rhs) const noexcept
- bool **operator>** (const type\_index &\_\_rhs) const noexcept
- bool **operator>=** (const type\_index &\_\_rhs) const noexcept

### 6.1007.1 Detailed Description

Class type\_index.

The class type\_index provides a simple wrapper for type\_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

The documentation for this struct was generated from the following file:

- [typeindex](#)

## 6.1008 \_\_gnu\_debug::type\_info Class Reference

```
#include <typeinfo>
```

### Public Member Functions

- virtual [~type\\_info](#) ()
- virtual bool **\_\_do\_catch** (const type\_info \*\_\_thr\_type, void \*\*\_\_thr\_obj, unsigned \_\_outer) const
- virtual bool **\_\_do\_upcast** (const \_\_cxxabiv1::\_\_class\_type\_info \*\_\_target, void \*\*\_\_obj\_ptr) const
- virtual bool **\_\_is\_function\_p** () const
- virtual bool **\_\_is\_pointer\_p** () const
- bool [before](#) (const type\_info &\_\_arg) const noexcept
- **size\_t hash\_code** () const noexcept
- const char \* [name](#) () const noexcept
- constexpr bool **operator==** (const type\_info &\_\_arg) const noexcept

### Protected Member Functions

- **type\_info** (const char \*\_\_n)

## Protected Attributes

- `const char * __name`

### 6.1008.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

### 6.1008.2 Constructor & Destructor Documentation

#### `~type_info()`

```
virtual std::type_info::~~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

### 6.1008.3 Member Function Documentation

#### `before()`

```
bool std::type_info::before (
 const type_info & __arg) const [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

#### `name()`

```
const char * std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

The documentation for this class was generated from the following file:

- `typeinfo`

## 6.1009 std::type\_info Class Reference

```
#include <typeinfo>
```

### Public Member Functions

- virtual `~type_info ()`
- virtual `bool __do_catch (const type_info * __thr_type, void ** __thr_obj, unsigned __outer) const`
- virtual `bool __do_upcast (const __cxxabiv1::__class_type_info * __target, void ** __obj_ptr) const`
- virtual `bool __is_function_p () const`
- virtual `bool __is_pointer_p () const`
- `bool before (const type_info & __arg) const noexcept`
- `size_t hash_code () const noexcept`
- `const char * name () const noexcept`
- `constexpr bool operator== (const type_info & __arg) const noexcept`

### Protected Member Functions

- `type_info (const char * __n)`

### Protected Attributes

- `const char * __name`

### 6.1009.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

### 6.1009.2 Constructor & Destructor Documentation

#### `~type_info()`

```
virtual std::type_info::~~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

### 6.1009.3 Member Function Documentation

#### `before()`

```
bool std::type_info::before (
 const type_info & __arg) const [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

#### `name()`

```
const char * std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 6.1010 std::type\_order< \_Tp, \_Up > Struct Template Reference

```
#include <compare>
```

### Public Types

- using **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr strong\_ordering **value**

### 6.1010.1 Detailed Description

```
template<typename _Tp, typename _Up>
```

```
struct std::type_order< _Tp, _Up >
```

Total ordering of types.

Since

C++26

The documentation for this struct was generated from the following file:

- [compare](#)

## 6.1011 `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`:



### Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ka::const_pointer` **key\_const\_pointer**
- typedef `__rebind_ka::const_reference` **key\_const\_reference**
- typedef `__rebind_ka::pointer` **key\_pointer**
- typedef `__rebind_ka::reference` **key\_reference**
- typedef `Key` **key\_type**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `Mapped` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `select_value_type< Key, Mapped >::type` **value\_type**

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### 6.1011.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >
```

Traits for abstract types.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 6.1012 std::chrono::tzdb\_list Class Reference

```
#include <chrono>
```

## Classes

- class [const\\_iterator](#)

## Public Member Functions

- **tzdb\_list** (const tzdb\_list &)=delete
- [const\\_iterator](#) **begin** () const noexcept
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_iterator](#) **cend** () const noexcept
- [const\\_iterator](#) **end** () const noexcept
- [const\\_iterator](#) **erase\_after** ([const\\_iterator](#) \_\_p)
- const tzdb & **front** () const noexcept
- tzdb\_list & **operator=** (const tzdb\_list &)=delete

## Friends

- bool **\_\_detail::\_\_recent\_leap\_second\_info** (leap\_second\_info &, unsigned)
- const tzdb & **get\_tzdb** ()
- tzdb\_list & **get\_tzdb\_list** ()
- class **leap\_second**
- const tzdb & **reload\_tzdb** ()
- struct **time\_zone::\_Impl**
- class **time\_zone\_link**
- struct **tzdb**

### 6.1012.1 Detailed Description

The list of `chrono::tzdb` objects

A single object of this type is constructed by the C++ runtime, and can be accessed by calling `chrono::get_tzdb_list()`.

The front of the list is the current `tzdb` object and can be accessed via `chrono::get_tzdb_list().front()` or `chrono::get_tzdb()` or `*chrono::get_tzdb_list().begin()`.

The `chrono::reload_tzdb()` function will check for a newer version and if found, insert it at the front of the list.

Since

C++20

### 6.1012.2 Member Function Documentation

#### erase\_after()

```
const_iterator std::chrono::tzdb_list::erase_after (
 const_iterator __p)
```

Remove the tzdb object *after* the one the iterator refers to.

Calling this function concurrently with any of `front()`, `begin()`, or `end()` does not cause a data race, but in general this function is not thread-safe. The behaviour may be undefined if erasing an element from the list while another thread is calling the same function, or incrementing an iterator into the list, or accessing the element being erased (unless it is accessed through an iterator).

#### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A dereferenceable iterator. |
|------------------|-----------------------------|

#### Returns

An iterator the element after the one that was erased (or `end()` if there is no such element).

#### Since

C++20

References [erase\\_after\(\)](#).

Referenced by [erase\\_after\(\)](#).

#### front()

```
const tzdb & std::chrono::tzdb_list::front () const [noexcept]
```

Access the current tzdb at the front of the list.

This returns a reference to the same object as `chrono::get_tzdb()`.

#### Returns

A reference to the current tzdb object.

#### Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

### 6.1013 `__gnu_cxx::unary_compose<_Operation1, _Operation2 >` Class Template Reference

```
#include <functional>
```

Inheritance diagram for `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`:



### Public Types

- typedef `_Operation2::argument_type` [argument\\_type](#)
- typedef `_Operation1::result_type` [result\\_type](#)

### Public Member Functions

- **unary\_compose** (const `_Operation1` &\_\_x, const `_Operation2` &\_\_y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &\_\_x) const

### Protected Attributes

- `_Operation1` **\_M\_fn1**
- `_Operation2` **\_M\_fn2**

#### 6.1013.1 Detailed Description

```
template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose< _Operation1, _Operation2 >
```

An [SGI extension](#) .

#### 6.1013.2 Member Typedef Documentation

##### **argument\_type**

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1::↵
result_type >::argument_type [inherited]
argument_type is the type of the argument
```

##### **result\_type**

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1::↵
result_type >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 6.1014 std::unary\_function<\_Arg, \_Result > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::unary\_function<\_Arg, \_Result >:



### Public Types

- typedef \_Arg [argument\\_type](#)
- typedef \_Result [result\\_type](#)

#### 6.1014.1 Detailed Description

```
template<typename _Arg, typename _Result>
struct std::unary_function<_Arg, _Result >
```

Helper for defining adaptable unary function objects.

**Deprecated** Deprecated in C++11, no longer in the standard since C++17.

#### 6.1014.2 Member Typedef Documentation

##### argument\_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function<_Arg, _Result >::argument_type
argument_type is the type of the argument
```

##### result\_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function<_Arg, _Result >::result_type
result_type is the return type
```

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



**6.1015 std::unary\_negate< \_Predicate > Class Template Reference**

```
#include <stl_function.h>
```

Inheritance diagram for std::unary\_negate< \_Predicate >:

**Public Types**

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

**Public Member Functions**

- constexpr **unary\_negate** (const `_Predicate` &\_\_x)
- constexpr `bool` **operator()** (const typename `_Predicate::argument_type` &\_\_x) const

**Protected Attributes**

- `_Predicate` **M\_pred**

**6.1015.1 Detailed Description**

```
template<typename _Predicate>
class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

**6.1015.2 Member Typedef Documentation****argument\_type**

```
typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::↔
argument_type [inherited]
argument_type is the type of the argument
```

**result\_type**

```
typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type [inherited]
result_type is the return type
```

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

**6.1016 \_\_gnu\_parallel::unbalanced\_tag Struct Reference**

```
#include <tags.h>
```

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:

**Public Member Functions**

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

**6.1016.1 Detailed Description**

Recommends parallel execution using static load-balancing at compile time.

**6.1016.2 Member Function Documentation****\_\_get\_num\_threads()**

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads() [inline], [inherited]
```

Find out desired number of threads.

**Returns**

Desired number of threads.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#), and [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\(\)](#).

**set\_num\_threads()**

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

**Parameters**

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

The documentation for this struct was generated from the following file:

- [tags.h](#)

**6.1017 std::underflow\_error Class Reference**

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::underflow\_error:

**Public Member Functions**

- **underflow\_error** (const char \*)
- **underflow\_error** (const [string](#) &\_\_arg)
- **underflow\_error** (const underflow\_error &)=default
- **underflow\_error** (underflow\_error &&)=default
- underflow\_error & **operator=** (const underflow\_error &)=default
- underflow\_error & **operator=** (underflow\_error &&)=default
- virtual const char \* **what** () const noexcept

**6.1017.1 Detailed Description**

Thrown to indicate arithmetic underflow.

**6.1017.2 Member Function Documentation****what()**

```
virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#), [std::filesystem::filesystem\\_error](#), and [std::ios\\_base::failure](#).

References [what\(\)](#).

Referenced by [what\(\)](#).

The documentation for this class was generated from the following file:

- [stdexcept\\_except.h](#)

## 6.1018 `std::underlying_type<_Tp>` Struct Template Reference

```
#include <type_traits>
```

### 6.1018.1 Detailed Description

```
template<typename _Tp>
struct std::underlying_type<_Tp>
```

The underlying type of an enum.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 6.1019 `std::uniform_int_distribution<_IntType>` Class Template Reference

```
#include <random>
```

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_IntType` [result\\_type](#)

### Public Member Functions

- [uniform\\_int\\_distribution](#) ()
- [uniform\\_int\\_distribution](#) (`_IntType __a, _IntType __b=__gnu_cxx::__int_traits<_IntType>::__max`)
- **`uniform_int_distribution`** (`const param\_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomBitGenerator>`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomBitGenerator>`  
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng, const param\_type &__p)`
- `template<typename _UniformRandomBitGenerator>`  
`void __generate (result\_type *__f, result\_type *__t, _UniformRandomBitGenerator &__urng, const param\_type &__p)`
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- `template<typename _UniformRandomBitGenerator>`  
`result\_type operator\(\) (_UniformRandomBitGenerator &__urng)`
- `template<typename _UniformRandomBitGenerator>`  
`result\_type operator\(\) (_UniformRandomBitGenerator &__urng, const param\_type &__p)`
- [param\\_type param](#) () const
- `void param (const param\_type &__param)`
- `void reset ()`

**Friends**

- bool `operator==` (const `uniform_int_distribution` &\_\_d1, const `uniform_int_distribution` &\_\_d2)

**6.1019.1 Detailed Description**

```
template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Since

C++11

**6.1019.2 Member Typedef Documentation****result\_type**

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

**6.1019.3 Constructor & Destructor Documentation****uniform\_int\_distribution() [1/2]**

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution () [inline]
```

Constructs a uniform distribution object.

**uniform\_int\_distribution() [2/2]**

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
 _IntType __a,
 _IntType __b = __gnu_cxx::__int_traits<_IntType>::__max) [inline], [explicit]
```

Constructs a uniform distribution object.

**6.1019.4 Member Function Documentation****max()**

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

**min()**

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

**operator()()**

```
template<typename _IntType = int>
template<typename _UniformRandomBitGenerator>
result_type std::uniform_int_distribution< _IntType >::operator() (
 _UniformRandomBitGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator\(\)\(\)](#).

**param() [1/2]**

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

**param() [2/2]**

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the uniform integer distribution.

**6.1019.5 Friends And Related Symbol Documentation****operator==**

```
template<typename _IntType = int>
bool operator== (
 const uniform_int_distribution< _IntType > & __d1,
 const uniform_int_distribution< _IntType > & __d2) [friend]
```

Return true if two uniform integer distributions have the same parameters.

The documentation for this class was generated from the following file:

- [uniform\\_int\\_dist.h](#)

**6.1020 std::uniform\_real\_distribution< \_RealType > Class Template Reference**

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- [uniform\\_real\\_distribution](#) ()
- [uniform\\_real\\_distribution](#) (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- [uniform\\_real\\_distribution](#) (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [a](#) () const
- [result\\_type](#) [b](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [uniform\\_real\\_distribution](#) &\_\_d1, const [uniform\\_real\\_distribution](#) &\_\_d2)

### 6.1020.1 Detailed Description

```
template<typename _RealType = double>
class std::uniform_real_distribution<_RealType>
```

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Since

C++11

### 6.1020.2 Member Typedef Documentation

#### [result\\_type](#)

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution<_RealType>::result_type
```

The type of the range of the distribution.

### 6.1020.3 Constructor & Destructor Documentation

#### uniform\_real\_distribution() [1/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution () [inline]
```

Constructs a `uniform_real_distribution` object.

The lower bound is set to 0.0 and the upper bound to 1.0

#### uniform\_real\_distribution() [2/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
 _RealType __a,
 _RealType __b = _RealType(1)) [inline], [explicit]
```

Constructs a `uniform_real_distribution` object.

#### Parameters

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__a</code> | [IN] The lower bound of the distribution. |
| <code>__b</code> | [IN] The upper bound of the distribution. |

### 6.1020.4 Member Function Documentation

#### max()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

#### min()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

#### operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::uniform_real_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator>\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).



**param()** [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution<_RealType>::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::uniform_real_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

Does nothing for the uniform real distribution.

**6.1020.5 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
 const uniform_real_distribution<_RealType> & __d1,
 const uniform_real_distribution<_RealType> & __d2) [friend]
```

Return true if two uniform real distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.1021 std::unique\_lock<\_Mutex> Class Template Reference**

```
#include <mutex>
```

**Public Types**

- typedef \_Mutex **mutex\_type**

**Public Member Functions**

- **unique\_lock** (const unique\_lock &)=delete
- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- template<typename \_Rep, typename \_Period>  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)<\_Rep, \_Period> &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
  **unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)<\_Clock, \_Duration> &\_\_atime)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (unique\_lock &&\_\_u) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept

- **operator bool** () const noexcept
- unique\_lock & **operator=** (const unique\_lock &)=delete
- unique\_lock & **operator=** (unique\_lock &&\_\_u) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** (unique\_lock &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period>  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### Related Symbols

(Note that these are not member symbols.)

- template<typename \_Mutex>  
void [swap](#) (unique\_lock< \_Mutex > &\_\_x, unique\_lock< \_Mutex > &\_\_y) noexcept

#### 6.1021.1 Detailed Description

**template<typename \_Mutex>**  
**class std::unique\_lock< \_Mutex >**

A movable scoped lock type.

A unique\_lock controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another unique\_lock by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Since

C++11

#### 6.1021.2 Friends And Related Symbol Documentation

##### swap()

```
template<typename _Mutex>
void swap (
 unique_lock< _Mutex > & __x,
 unique_lock< _Mutex > & __y) [related]
```

Swap overload for unique\_lock objects.

The documentation for this class was generated from the following file:

- [unique\\_lock.h](#)

### 6.1022 std::unique\_ptr< \_Tp, \_Dp > Class Template Reference

```
#include <memory>
```

#### Public Types

- using **deleter\_type**
- using **element\_type**
- using **pointer**

## Public Member Functions

- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
constexpr [unique\\_ptr](#) () noexcept
- template<typename \_Up, typename = \_Require<is\_convertible<\_Up\*, pointer>, is\_same<\_Dp, default\_delete<\_Tp>>>>  
[unique\\_ptr](#) (auto\_ptr<\_Up> &&\_\_u) noexcept
- [unique\\_ptr](#) (const unique\_ptr &)=delete
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
constexpr [unique\\_ptr](#) (nullptr\_t) noexcept
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
constexpr [unique\\_ptr](#) (pointer \_\_p) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_move\_constructible<\_Del>>>  
constexpr [unique\\_ptr](#) (pointer \_\_p, \_\_enable\_if\_t<!is\_lvalue\_reference<\_Del>::value, \_Del && > \_\_d) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_copy\_constructible<\_Del>>>  
constexpr [unique\\_ptr](#) (pointer \_\_p, const deleter\_type &\_\_d) noexcept
- template<typename \_Del = deleter\_type, typename \_DelUnref = typename remove\_reference<\_Del>::type>  
constexpr [unique\\_ptr](#) (pointer, \_\_enable\_if\_t<is\_lvalue\_reference<\_Del>::value, \_DelUnref && >)=delete
- [unique\\_ptr](#) (unique\_ptr &&)=default
- template<typename \_Up, typename \_Ep, typename = \_Require<\_\_safe\_conversion\_up<\_Up, \_Ep>, \_\_conditional\_t<is\_reference<\_Up>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>>>  
constexpr [unique\\_ptr](#) (unique\_ptr<\_Up, \_Ep> &&\_\_u) noexcept
- constexpr ~[unique\\_ptr](#) () noexcept
- constexpr pointer [get](#) () const noexcept
- constexpr const deleter\_type & [get\\_deleter](#) () const noexcept
- constexpr deleter\_type & [get\\_deleter](#) () noexcept
- constexpr [operator bool](#) () const noexcept
- constexpr [add\\_lvalue\\_reference](#)< element\_type >::type [operator\\*](#) () const noexcept(noexcept(\*std::declval<pointer>()))
- constexpr pointer [operator->](#) () const noexcept
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- constexpr [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&)=default
- template<typename \_Up, typename \_Ep>  
constexpr [enable\\_if](#)<\_\_and<\_\_safe\_conversion\_up<\_Up, \_Ep>, [is\\_assignable](#)< deleter\_type &, \_Ep && >  
>::value, [unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#)<\_Up, \_Ep> &&\_\_u) noexcept
- constexpr pointer [release](#) () noexcept
- constexpr void [reset](#) (pointer \_\_p=pointer()) noexcept
- constexpr void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept

## Friends

- template<typename, typename, typename...>  
class [inout\\_ptr\\_t](#)
- template<typename, typename, typename...>  
class [out\\_ptr\\_t](#)

## Related Symbols

(Note that these are not member symbols.)

- template<typename \_CharT, typename \_Traits, typename \_Tp, typename \_Dp>  
[basic\\_ostream](#)<\_CharT, \_Traits> & [operator<<](#) ([basic\\_ostream](#)<\_CharT, \_Traits> &\_\_os, const [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_p)

- `template<typename _Tp, typename _Dp>`  
`constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x,`  
`unique_ptr< _Tp, _Dp > &__y) noexcept`

### 6.1022.1 Detailed Description

`template<typename _Tp, typename _Dp = default_delete<_Tp>>`  
`class std::unique_ptr< _Tp, _Dp >`

A move-only smart pointer that manages unique ownership of a resource.

Since

C++11

### 6.1022.2 Constructor & Destructor Documentation

#### `unique_ptr()` [1/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr () [inline], [constexpr], [noexcept]
```

Default constructor, creates a `unique_ptr` that owns nothing.

#### `unique_ptr()` [2/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p) [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__p</code> | A pointer to an object of <code>element_type</code> |
|------------------|-----------------------------------------------------|

The deleter will be value-initialized.

#### `unique_ptr()` [3/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p,
 const deleter_type & __d) [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__p</code> | A pointer to an object of <code>element_type</code> |
| <code>__d</code> | A reference to a deleter.                           |

The deleter will be initialized with `__d`

#### unique\_ptr() [4/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p,
 __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__p</code> | A pointer to an object of <code>element_type</code> |
| <code>__d</code> | An rvalue reference to a (non-reference) deleter.   |

The deleter will be initialized with `std::move(__d)`

#### unique\_ptr() [5/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 nullptr_t) [inline], [constexpr], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

#### unique\_ptr() [6/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 unique_ptr< _Tp, _Dp > &&) [default]
```

Move constructor.

#### unique\_ptr() [7/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __
conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Converting constructor from another type.

Requires that the pointer owned by `__u` is convertible to the type of pointer owned by this object, `__u` does not own an array, and `__u` has a compatible deleter type.

#### unique\_ptr() [8/8]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 auto_ptr< _Up > && __u) [inline], [noexcept]
```

Converting constructor from `auto_ptr`.

**~unique\_ptr()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::~~unique_ptr () [inline], [constexpr], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

**6.1022.3 Member Function Documentation****get()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::get () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

Referenced by [std::unique\\_ptr<\\_State>::operator bool\(\)](#), [std::unique\\_ptr<\\_State>::operator->\(\)](#), [std::operator<\(\)>](#), [std::operator<\(\)>](#), [std::operator<\(\)>](#), [std::unique\\_ptr<\\_State>::operator<<\(\)](#), [std::operator==\(\)](#), [std::operator>\(\)](#), and [std::operator>\(\)](#).

**get\_deleter() [1/2]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () const [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

**get\_deleter() [2/2]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

Referenced by [std::unique\\_ptr<\\_State>::~~unique\\_ptr\(\)](#), and [std::unique\\_ptr<\\_State>::operator=\(\)](#).

**operator bool()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::operator bool () const [inline], [explicit], [constexpr], [noexcept]
```

Return true if the stored pointer is not null.

**operator\*()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

**operator->()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

**operator=() [1/3]**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
 nullptr_t) [inline], [constexpr], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

**operator=()** [2/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr<_Tp, _Dp>::operator= (
 unique_ptr<_Tp, _Dp> &&) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

**operator=()** [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep>
enable_if< __and< __safe_conversion_up<_Up, _Ep>, is_assignable< deleter_type &, _Ep && >
::value, unique_ptr & >::type std::unique_ptr<_Tp, _Dp>::operator= (
 unique_ptr<_Up, _Ep> && __u) [inline], [constexpr], [noexcept]
```

Assignment from another type.

**Parameters**

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| $\leftarrow$<br>_U | The object to transfer ownership from, which owns a convertible pointer to a non-array object. |
|--------------------|------------------------------------------------------------------------------------------------|

Invokes the deleter if this object owns a pointer.

**release()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::release () [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

**reset()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::reset (
 pointer __p = pointer()) [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

**Parameters**

|                    |                           |
|--------------------|---------------------------|
| $\leftarrow$<br>_p | The new pointer to store. |
|--------------------|---------------------------|

The deleter will be invoked if a pointer is already owned.

Referenced by [std::unique\\_ptr<\\_State>::operator=\(\)](#), and [std::unique\\_ptr<\\_State>::operator=\(\)](#).

**swap()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::swap (
 unique_ptr<_Tp, _Dp> & __u) [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 6.1023 `std::unique_ptr<_Tp[],_Dp>` Class Template Reference

```
#include <memory>
```

### Public Types

- `template<typename _Up>`  
using `__safe_conversion_raw`
- `template<typename _Up, typename _Ep, typename _UPtr = unique_ptr<_Up, _Ep>, typename _UP_pointer = typename _UPtr::pointer,`  
`typename _UP_element_type = typename _UPtr::element_type>`  
using `__safe_conversion_up`
- using `deleter_type`
- using `element_type`
- using `pointer`

### Public Member Functions

- `template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>`  
`constexpr unique_ptr () noexcept`
- `template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename enable_if< __safe_↵`  
`conversion_raw<_Up>::value, bool>::type>`  
`constexpr unique_ptr (_Up __p) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_move_↵`  
`constructible<_Del>>>`  
`constexpr unique_ptr (_Up __p, __enable_if_t<!is_lvalue_reference<_Del>::value, _Del && > __d) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_copy_constructible<↵`  
`_Del>>>`  
`constexpr unique_ptr (_Up __p, const deleter_type & __d) noexcept`
- `template<typename _Up, typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type, typename =`  
`_Require<__safe_conversion_raw<_Up>>>`  
`unique_ptr (_Up, __enable_if_t< is_lvalue_reference<_Del>::value, _DelUnref && >)=delete`
- `unique_ptr (const unique_ptr &)=delete`
- `template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>`  
`constexpr unique_ptr (nullptr_t) noexcept`
- `unique_ptr (unique_ptr &&)=default`
- `template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<↵`  
`_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>`  
`constexpr unique_ptr (unique_ptr<_Up, _Ep> && __u) noexcept`
- `constexpr ~unique_ptr ()`
- `constexpr pointer get () const noexcept`
- `constexpr const deleter_type & get_deleter () const noexcept`
- `constexpr deleter_type & get_deleter () noexcept`
- `constexpr operator bool () const noexcept`
- `constexpr add_lvalue_reference< element_type >::type operator* () const noexcept(noexcept(*std::declval<`  
`pointer >()))`
- `constexpr pointer operator-> () const noexcept`
- `unique_ptr & operator= (const unique_ptr &)=delete`
- `constexpr unique_ptr & operator= (nullptr_t) noexcept`
- `unique_ptr & operator= (unique_ptr &&)=default`
- `template<typename _Up, typename _Ep>`  
`constexpr enable_if< __and< __safe_conversion_up<_Up, _Ep>, is_assignable< deleter_type &, _Ep && >`  
`>::value, unique_ptr & >::type operator= (unique_ptr<_Up, _Ep> && __u) noexcept`
- `constexpr std::add_lvalue_reference< element_type >::type operator[] (size_t __i) const`



- constexpr pointer [release](#) () noexcept
- template<typename \_Up, typename = \_Require< \_\_or\_<is\_same<\_Up, pointer>, \_\_and\_<is\_same<pointer, element\_type\*>, is\_↵ pointer<\_Up>, is\_convertible< typename remove\_pointer<\_Up>::type(\*)[], element\_type(\*)[] > > > >>> constexpr void [reset](#) (\_Up \_\_p) noexcept
- constexpr void [reset](#) (nullptr\_t=nullptr) noexcept
- constexpr void [swap](#) (unique\_ptr &\_\_u) noexcept

## Friends

- template<typename, typename, typename...> class [inout\\_ptr\\_t](#)
- template<typename, typename, typename...> class [out\\_ptr\\_t](#)

## Related Symbols

(Note that these are not member symbols.)

- [basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_p)
- constexpr [enable\\_if](#)< \_\_is\_swappable< \_Dp >::value >::type [swap](#) ([unique\\_ptr](#)< \_Tp, \_Dp > &\_\_↵ x, [unique\\_ptr](#)< \_Tp, \_Dp > &\_\_y) noexcept

### 6.1023.1 Detailed Description

**template<typename \_Tp, typename \_Dp>**  
**class std::unique\_ptr< \_Tp[], \_Dp >**

A move-only smart pointer that manages unique ownership of an array.

Since

C++11

### 6.1023.2 Constructor & Destructor Documentation

#### [unique\\_ptr\(\)](#) [1/6]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr () [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique\_ptr that owns nothing.

References [unique\\_ptr\(\)](#).

Referenced by [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [unique\\_ptr\(\)](#), [~unique\\_ptr\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [swap\(\)](#).

#### [unique\\_ptr\(\)](#) [2/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p) [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

**Parameters**

|                                                                                                                      |                                                                                             |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a><br><code>_p</code> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|

The deleter will be value-initialized.

References [unique\\_ptr\(\)](#).

**unique\_ptr()** [3/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<
_Up>, is_copy_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p,
 const deleter_type & __d) [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

**Parameters**

|                                                                                                                      |                                                                                             |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a><br><code>_p</code> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
| <a href="#"></a><br><code>_d</code> | A reference to a deleter.                                                                   |

The deleter will be initialized with `__d`

References [unique\\_ptr\(\)](#).

**unique\_ptr()** [4/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<
_Up>, is_move_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p,
 __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

**Parameters**

|                                                                                                                        |                                                                                             |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#"></a><br><code>_p</code> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
| <a href="#"></a><br><code>_d</code> | A reference to a deleter.                                                                   |

The deleter will be initialized with `std::move(__d)`

References [unique\\_ptr\(\)](#), and [std::move\(\)](#).

**unique\_ptr()** [5/6]

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
```

```
unique_ptr< _Tp[], _Dp > &&) [default]
```

Move constructor.

References [unique\\_ptr\(\)](#).

### **unique\_ptr()** [6/6]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 nullptr_t) [inline], [constexpr], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

References [unique\\_ptr\(\)](#).

### **~unique\_ptr()**

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::~~unique_ptr () [inline], [constexpr]
```

Destructor, invokes the deleter if the stored pointer is not null.

References [unique\\_ptr\(\)](#), and [get\\_deleter\(\)](#).

## **6.1023.3 Member Function Documentation**

### **get()**

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::get () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

References [get\(\)](#).

Referenced by [get\(\)](#), [operator bool\(\)](#), and [operator\[\]\(\)](#).

### **get\_deleter()** [1/2]

```
template<typename _Tp, typename _Dp>
const deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter () const [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

References [get\\_deleter\(\)](#).

### **get\_deleter()** [2/2]

```
template<typename _Tp, typename _Dp>
deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter () [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

References [get\\_deleter\(\)](#).

Referenced by [~unique\\_ptr\(\)](#), [get\\_deleter\(\)](#), [get\\_deleter\(\)](#), and [operator=\(\)](#).

### **operator bool()**

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::operator bool () const [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.

References [get\(\)](#).

**operator\*()**

```
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

**operator->()**

```
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const [inline], [constexpr], [noexcept]
```

Return the stored pointer.

**operator=() [1/3]**

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
 nullptr_t) [inline], [constexpr], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

References [unique\\_ptr\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**operator=() [2/3]**

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
 unique_ptr< _Tp[], _Dp > &&) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

References [unique\\_ptr\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**operator=() [3/3]**

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Ep>
enable_if< __and< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp[], _Dp >::operator= (
 unique_ptr< _Up, _Ep > && __u) [inline], [constexpr], [noexcept]
```

Assignment from another type.

**Parameters**

|                    |                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------|
| <a href="#">_↵</a> | The object to transfer ownership from, which owns a convertible pointer to an array object. |
| <a href="#">_U</a> |                                                                                             |

Invokes the deleter if this object owns a pointer.

References [unique\\_ptr\(\)](#), [std::forward\(\)](#), [get\\_deleter\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**operator[]()**

```
template<typename _Tp, typename _Dp>
std::add_lvalue_reference< element_type >::type std::unique_ptr< _Tp[], _Dp >::operator[] (
 size_t __i) const [inline], [constexpr]
```

Access an element of owned array.

References [get\(\)](#), and [operator\[\]\(\)](#).

Referenced by [operator\[\]\(\)](#).

**release()**

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr<_Tp[], _Dp>::release () [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

References [release\(\)](#).

Referenced by [release\(\)](#).

**reset()**

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element_↵
_type(*)[] > > > >>
void std::unique_ptr<_Tp[], _Dp>::reset (
 _Up __p) [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

**Parameters**

|       |                           |
|-------|---------------------------|
| $\_p$ | The new pointer to store. |
|-------|---------------------------|

The deleter will be invoked if a pointer is already owned.

References [std::move\(\)](#), and [reset\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), and [reset\(\)](#).

**swap()**

```
template<typename _Tp, typename _Dp>
void std::unique_ptr<_Tp[], _Dp>::swap (
 unique_ptr<_Tp[], _Dp> & __u) [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

References [unique\\_ptr\(\)](#), and [swap\(\)](#).

Referenced by [swap\(\)](#).

**6.1023.4 Friends And Related Symbol Documentation****operator<<()**

```
basic_ostream<_CharT, _Traits> & operator<< (
 basic_ostream<_CharT, _Traits> & __os,
 const unique_ptr<_Tp, _Dp> & __p) [related]
```

Stream output operator for unique\_ptr.

Since

C++20

**swap()**

```
enable_if< __is_swappable<_Dp>::value >::type swap (
 unique_ptr<_Tp, _Dp> & __x,
 unique_ptr<_Tp, _Dp> & __y) [related]
```

Swap overload for unique\_ptr.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 6.1024 std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

Inheritance diagram for std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >:



### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, unordered\_map > **const\_iterator**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_const\_local\_iterator, unordered\_map > **const\_local\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, unordered\_map > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_local\_iterator, unordered\_map > **local\_iterator**
- typedef \_Base::mapped\_type **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

### Public Member Functions

- **unordered\_map** ( \_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**unordered\_map** ( \_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** ( \_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** ( \_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** ( \_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())

- **unordered\_map** (const allocator\_type &\_\_a)
- **unordered\_map** (const unordered\_map &)=default
- **unordered\_map** (const unordered\_map &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (unordered\_map &&)=default
- **unordered\_map** (unordered\_map &&\_\_umap, const allocator\_type &\_\_a) noexcept(noexcept(\_Base(std::move(\_\_umap), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local\_iterator **begin** (size\_type \_\_b)
- const\_local\_iterator **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const\_iterator **cbegin** () const noexcept
- const\_local\_iterator **cbegin** (size\_type \_\_b) const
- const\_iterator **cend** () const noexcept
- const\_local\_iterator **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
std::pair< iterator, bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- local\_iterator **end** (size\_type \_\_b)
- const\_local\_iterator **end** (size\_type \_\_b) const
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_it)
- template<\_\_heterogeneous\_hash\_key< unordered\_map > \_Kt>  
size\_type **erase** (\_Kt &&\_\_key)
- size\_type **erase** (const key\_type &\_\_key)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_it)
- iterator **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
std::pair< iterator, bool > **insert** (\_Pair &&\_\_obj)

- [std::pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type> iterator **insert** (const\_iterator \_\_hint, \_Pair &&\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, value\_type &&\_\_x)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (value\_type &&\_\_x)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- unordered\_map & **operator=** (const unordered\_map &)=default
- unordered\_map & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- unordered\_map & **operator=** (unordered\_map &&)=default
- void **swap** (unordered\_map &\_\_x) noexcept(noexcept([declval](#)< \_Base & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)
- template<typename \_ItT, typename \_SeqT>  
class ::[\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)

#### 6.1024.1 Detailed Description

**template<typename \_Key, typename \_Tp, typename \_Hash = std::hash<\_Key>, typename \_Pred = std::equal<\_to<\_Key>, typename \_Alloc = std::allocator<std::pair<const \_Key, \_Tp> >>  
class std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >**

Class std::unordered\_map with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

#### 6.1025 std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::mapped\_type [mapped\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)



- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_map](#) ()=default
- template<typename \_InputIterator>  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator>  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_map](#) (const unordered\_map &)=default
- **unordered\_map** (const unordered\_map &\_\_umap, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** (initializer\_list< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** (initializer\_list< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_map** ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- **unordered\_map** ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_map](#) (unordered\_map &&)=default
- **unordered\_map** (unordered\_map &&\_\_umap, const [allocator\\_type](#) &\_\_a) noexcept(noexcept(\_Hashtable(std::move(↵\_\_umap.\_M\_h), \_\_a)))
- [iterator](#) [begin](#) () noexcept
- [local\\_iterator](#) [begin](#) ([size\\_type](#) \_\_n)
- [size\\_type](#) [bucket\\_count](#) () const noexcept
- [size\\_type](#) [bucket\\_size](#) ([size\\_type](#) \_\_n) const
- void [clear](#) () noexcept
- template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
- bool [empty](#) () const noexcept
- [iterator](#) [end](#) () noexcept
- [local\\_iterator](#) [end](#) ([size\\_type](#) \_\_n)
- [iterator](#) [erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type](#) [get\\_allocator](#) () const noexcept
- [hasher](#) [hash\\_function](#) () const

- `template<typename _InputIterator>`  
`void insert (_InputIterator __first, _InputIterator __last)`
  - `void insert (initializer_list< value_type > __l)`
  - `key_equal key_eq () const`
  - `float load_factor () const noexcept`
  - `size_type max_bucket_count () const noexcept`
  - `float max_load_factor () const noexcept`
  - `void max_load_factor (float __z)`
  - `size_type max_size () const noexcept`
  - `unordered_map & operator= (const unordered_map &)=default`
  - `unordered_map & operator= (initializer_list< value_type > __l)`
  - `unordered_map & operator= (unordered_map &&)=default`
  - `void rehash (size_type __n)`
  - `void reserve (size_type __n)`
  - `size_type size () const noexcept`
  - `void swap (unordered_map & __x) noexcept(noexcept(_M_h.swap(__x._M_h)))`
- 
- `const_iterator begin () const noexcept`
  - `const_iterator cbegin () const noexcept`
- 
- `const_iterator end () const noexcept`
  - `const_iterator cend () const noexcept`
- 
- `std::pair< iterator, bool > insert (const value_type & __x)`
  - `std::pair< iterator, bool > insert (value_type && __x)`
  - `template<typename _Pair>`  
`__enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > insert (_Pair && __x)`
- 
- `iterator insert (const_iterator __hint, const value_type & __x)`
  - `iterator insert (const_iterator __hint, value_type && __x)`
  - `template<typename _Pair>`  
`__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __hint, _Pair && __x)`
- 
- `iterator erase (const_iterator __position)`
  - `iterator erase (iterator __position)`
- 
- `size_type erase (const key_type & __x)`
  - `template<__heterogeneous_hash_key< unordered_map > _Kt>`  
`size_type erase (_Kt && __x)`
- 
- `iterator find (const key_type & __x)`
  - `const_iterator find (const key_type & __x) const`

- `size_type count` (const `key_type` &\_\_x) const
- `bool contains` (const `key_type` &\_\_x) const
- `template<typename _Kt>`  
`auto contains` (const `_Kt` &\_\_x) const -> `decltype(_M_h._M_find_tr(__x), void(), true)`
- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const
- `mapped_type & operator[]` (const `key_type` &\_\_k)
- `mapped_type & operator[]` (`key_type` &&\_\_k)
- `template<__heterogeneous_hash_key< unordered_map > _Kt>`  
`mapped_type & operator[]` (`_Kt` &&\_\_k)
- `mapped_type & at` (const `key_type` &\_\_k)
- `template<__heterogeneous_hash_key< unordered_map > _Kt>`  
`mapped_type & at` (const `_Kt` &\_\_k)
- `const mapped_type & at` (const `key_type` &\_\_k) const
- `template<__heterogeneous_hash_key< unordered_map > _Kt>`  
`const mapped_type & at` (const `_Kt` &\_\_k) const
- `size_type bucket` (const `key_type` &\_\_key) const
- `template<__heterogeneous_hash_key< unordered_map > _Kt>`  
`size_type bucket` (const `_Kt` &\_\_key) const
- `const_local_iterator begin` (`size_type` \_\_n) const
- `const_local_iterator cbegin` (`size_type` \_\_n) const
- `const_local_iterator end` (`size_type` \_\_n) const
- `const_local_iterator cend` (`size_type` \_\_n) const

## Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>`  
`bool operator==` (const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

### 6.1025.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Since

C++11

#### Template Parameters

|                     |                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                                                  |
| <code>_Tp</code>    | Type of mapped objects.                                                                               |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Key&gt;</code> .                             |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Key&gt;</code> .                       |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;&gt;</code> . |

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

### 6.1025.2 Member Typedef Documentation

#### `allocator_type`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

#### `const_iterator`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

#### `const_local_iterator`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

#### `const_pointer`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
typedef _Hashtable::const_pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::const_↵
pointer
Iterator-related typedefs.
```

### const\_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::↵
const_reference
Iterator-related typedefs.
```

### difference\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::↵
difference_type
Iterator-related typedefs.
```

### hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::hasher
Public typedefs.
```

### iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::iterator
Iterator-related typedefs.
```

### key\_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

### key\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

### local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::local_↵
_iterator
Iterator-related typedefs.
```

**mapped\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::mapped_type
Public typedefs.
```

**pointer**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer
Iterator-related typedefs.
```

**reference**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::value_type
Public typedefs.
```

**6.1025.3 Constructor & Destructor Documentation****unordered\_map() [1/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>() [default]
Default constructor.
```

**unordered\_map() [2/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

**unordered\_map()** [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_map from a range.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_map consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first, __last)`).

**unordered\_map()** [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

**unordered\_map()** [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>
```

```
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

### unordered\_map() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_map with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

### unordered\_map() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_map from an initializer\_list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_map consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

## 6.1025.4 Member Function Documentation

### at() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>
```



```
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
 const _Kt & __k) [inline]
```

Access to unordered\_map data.

#### Parameters

|                       |                                             |
|-----------------------|---------------------------------------------|
| <code>↔<br/>_k</code> | The key for which data should be retrieved. |
|-----------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the unordered\_map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

#### at() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
 const _Kt & __k) const [inline]
```

Access to unordered\_map data.

#### Parameters

|                       |                                             |
|-----------------------|---------------------------------------------|
| <code>↔<br/>_k</code> | The key for which data should be retrieved. |
|-----------------------|---------------------------------------------|

#### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the unordered\_map.

#### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

#### at() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
 const key_type & __k) [inline]
```

Access to unordered\_map data.

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <code>_↵<br/>_k</code> | The key for which data should be retrieved. |
|------------------------|---------------------------------------------|

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If no such data is present. |
|------------------------------------------------|-----------------------------|

**at()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
 const key_type & __k) const [inline]
```

Access to `unordered_map` data.

**Parameters**

|                        |                                             |
|------------------------|---------------------------------------------|
| <code>_↵<br/>_k</code> | The key for which data should be retrieved. |
|------------------------|---------------------------------------------|

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

|                                                |                             |
|------------------------------------------------|-----------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If no such data is present. |
|------------------------------------------------|-----------------------------|

**begin()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

**begin()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

**begin()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read/write local iterator.

**begin()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_map.

**cbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

**cbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**cend()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

**cend()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
 const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__x</code> | Key to count. |
|------------------|---------------|

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multimap; for unordered\_map the result will either be 0 (not present) or 1 (present).

**emplace()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`. Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**empty()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered\_map is empty.

#### end() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

#### end() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered\_map.

#### end() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

#### Parameters

|       |                   |
|-------|-------------------|
| $\_↔$ | The bucket index. |
| $\_n$ |                   |

#### Returns

A read/write local iterator.

#### end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|       |                   |
|-------|-------------------|
| $\_↔$ | The bucket index. |
| $\_n$ |                   |

#### Returns

A read-only local iterator.

**equal\_range()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

**equal\_range()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

**erase()** [1/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 _Kt && __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|



**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_map. For an unordered\_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_map. For an unordered\_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_map.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an unordered\_map. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [5/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an unordered\_map.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↔</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered\_map.

**hash\_function()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_map was constructed.

**insert() [1/8]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

#### insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

#### Parameters

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

#### insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

#### Parameters

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 _Pair && __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_map`.

**Parameters**

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
|------------------|------------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <code>_↔</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
| <code>_X</code> |                                                                      |

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_map was constructed.

**load\_factor()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered\_map.

**max\_load\_factor() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_map tries to keep the load factor less than or equal to.

**max\_load\_factor() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
```

```
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
float __z) [inline]
```

Change the unordered\_map maximum load factor.

**Parameters**

|                 |                              |
|-----------------|------------------------------|
| <code>_↵</code> | The new maximum load factor. |
| <code>_Z</code> |                              |

**max\_size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_map.

**operator=()** [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_map list assignment operator.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| <code>↵</code>  | An initializer_list. |
| <code>_↵</code> |                      |
| <code>↵</code>  |                      |
| <code>_↵</code> |                      |
| <code>/</code>  |                      |

This function fills an unordered\_map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

**operator[]()** [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```



```
template<__heterogeneous_hash_key< unordered_map > _Kt>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
 _Kt && __k) [inline]
```

Subscript ( [] ) access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned. Lookup requires constant time.

#### operator[]() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
 const key_type & __k) [inline]
```

Subscript ( [] ) access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

#### Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned. Lookup requires constant time.

#### operator[]() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
 key_type && __k) [inline]
```

Subscript ( [] ) access to unordered\_map data.

#### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` )operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned. Lookup requires constant time.

**rehash()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_map.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_map maximum load factor.

**reserve()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_map for a specified number of elements.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

**size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the unordered\_map.

**swap()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_map.

**Parameters**

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| <code>_↔</code> | An unordered_map of the same element and allocator types. |
| <code>_X</code> |                                                           |

This exchanges the elements between two unordered\_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 6.1026 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

Inheritance diagram for std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >:

**Public Types**

- typedef \_Base::allocator\_type **allocator\_type**
- typedef [\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, unordered\_multimap > **const\_iterator**
- typedef [\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_const\_local\_iterator, unordered\_multimap > **const\_local\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef [\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, unordered\_multimap > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef [\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_local\_iterator, unordered\_multimap > **local\_iterator**
- typedef \_Base::mapped\_type **mapped\_type**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

**Public Member Functions**

- **unordered\_multimap** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)

- `template<typename _InputIterator>`  
`unordered_multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `size_type` \_\_n, `const allocator_type` &\_\_a)
- `template<typename _InputIterator>`  
`unordered_multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `size_type` \_\_n, `const hasher` &\_\_hf, `const allocator_type` &\_\_a)
- `template<typename _InputIterator>`  
`unordered_multimap` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `size_type` \_\_n=0, `const hasher` &\_\_hf=hasher(), `const key_equal` &\_\_eq=key\_equal(), `const allocator_type` &\_\_a=allocator\_type())
- `unordered_multimap` (`const allocator_type` &\_\_a)
- `unordered_multimap` (`const unordered_multimap` &)=default
- `unordered_multimap` (`const unordered_multimap` &\_\_umap, `const allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `const allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, `const allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, `const hasher` &\_\_hf, `const allocator_type` &\_\_a)
- `unordered_multimap` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, `const hasher` &\_\_hf=hasher(), `const key_equal` &\_\_eq=key\_equal(), `const allocator_type` &\_\_a=allocator\_type())
- `unordered_multimap` (`size_type` \_\_n, `const allocator_type` &\_\_a)
- `unordered_multimap` (`size_type` \_\_n, `const hasher` &\_\_hf, `const allocator_type` &\_\_a)
- `unordered_multimap` (`size_type` \_\_n, `const hasher` &\_\_hf=hasher(), `const key_equal` &\_\_eq=key\_equal(), `const allocator_type` &\_\_a=allocator\_type())
- `unordered_multimap` (`unordered_multimap` &&)=default
- `unordered_multimap` (`unordered_multimap` &&\_\_umap, `const allocator_type` &\_\_a) noexcept(noexcept(`Base`(`std::move`(\_\_umap), \_\_a)))
- `const _Base` & `_M_base` () `const` noexcept
- `_Base` & `_M_base` () noexcept
- `const_iterator` `begin` () `const` noexcept
- `iterator` `begin` () noexcept
- `local_iterator` `begin` (`size_type` \_\_b)
- `const_local_iterator` `begin` (`size_type` \_\_b) `const`
- `size_type` `bucket_size` (`size_type` \_\_b) `const`
- `const_iterator` `cbegin` () `const` noexcept
- `const_local_iterator` `cbegin` (`size_type` \_\_b) `const`
- `const_iterator` `cend` () `const` noexcept
- `const_local_iterator` `cend` (`size_type` \_\_b) `const`
- `void` `clear` () noexcept
- `template<typename... _Args>`  
`iterator` `emplace` (`_Args` &&... \_\_args)
- `template<typename... _Args>`  
`iterator` `emplace_hint` (`const_iterator` \_\_hint, `_Args` &&... \_\_args)
- `const_iterator` `end` () `const` noexcept
- `iterator` `end` () noexcept
- `local_iterator` `end` (`size_type` \_\_b)
- `const_local_iterator` `end` (`size_type` \_\_b) `const`
- `std::pair`< `iterator`, `iterator` > `equal_range` (`const key_type` &\_\_key)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (`const key_type` &\_\_key) `const`
- `_Base_iterator` `erase` (`_Base_const_iterator` \_\_it)
- `template<_heterogeneous_hash_key< unordered_multimap > _Kt>`  
`size_type` `erase` (`_Kt` &&\_\_key)
- `size_type` `erase` (`const key_type` &\_\_key)
- `iterator` `erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)

- iterator **erase** (const\_iterator \_\_it)
- iterator **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (\_Pair &&\_\_obj)
- iterator **insert** (const value\_type &\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_hint, \_Pair &&\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, value\_type &&\_\_x)
- void **insert** (std::initializer\_list< value\_type > \_\_l)
- iterator **insert** (value\_type &&\_\_x)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- unordered\_multimap & **operator=** (const unordered\_multimap &)=default
- unordered\_multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- unordered\_multimap & **operator=** (unordered\_multimap &&)=default
- void **swap** (unordered\_multimap &\_\_x) noexcept(noexcept(declval< \_Base & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::\_\_gnu\_debug::Safe\_iterator
- template<typename \_ItT, typename \_SeqT>  
class ::\_\_gnu\_debug::Safe\_local\_iterator

### 6.1026.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_↵
_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class std::unordered\_multimap with safety/checking/debug instrumentation.  
The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 6.1027 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_map>
```

## Public Types

- typedef `_Hashtable::key_type` [key\\_type](#)
  - typedef `_Hashtable::value_type` [value\\_type](#)
  - typedef `_Hashtable::mapped_type` [mapped\\_type](#)
  - typedef `_Hashtable::hasher` [hasher](#)
  - typedef `_Hashtable::key_equal` [key\\_equal](#)
  - typedef `_Hashtable::allocator_type` [allocator\\_type](#)
- 
- typedef `_Hashtable::pointer` [pointer](#)
  - typedef `_Hashtable::const_pointer` [const\\_pointer](#)
  - typedef `_Hashtable::reference` [reference](#)
  - typedef `_Hashtable::const_reference` [const\\_reference](#)
  - typedef `_Hashtable::iterator` [iterator](#)
  - typedef `_Hashtable::const_iterator` [const\\_iterator](#)
  - typedef `_Hashtable::local_iterator` [local\\_iterator](#)
  - typedef `_Hashtable::const_local_iterator` [const\\_local\\_iterator](#)
  - typedef `_Hashtable::size_type` [size\\_type](#)
  - typedef `_Hashtable::difference_type` [difference\\_type](#)

## Public Member Functions

- [unordered\\_multimap](#) ()=default
- template<typename `_InputIterator`>  
**unordered\_multimap** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
**unordered\_multimap** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
**unordered\_multimap** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename `_InputIterator`>  
[unordered\\_multimap](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multimap](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_multimap](#) (const `unordered_multimap` &)=default
- **unordered\_multimap** (const `unordered_multimap` &\_\_ummap, const [allocator\\_type](#) &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multimap](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **unordered\_multimap** ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- **unordered\_multimap** ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multimap](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multimap](#) (`unordered_multimap` &&)=default
- **unordered\_multimap** (`unordered_multimap` &&\_\_ummap, const [allocator\\_type](#) &\_\_a) noexcept(noexcept(`std::move`(\_\_ummap.\_M\_h), \_\_a))
- [iterator](#) [begin](#) () noexcept

- [local\\_iterator begin](#) (size\_type \_\_n)
  - [size\\_type bucket\\_count](#) () const noexcept
  - [size\\_type bucket\\_size](#) (size\_type \_\_n) const
  - void [clear](#) () noexcept
  - template<typename... \_Args>  
[iterator emplace](#) (\_Args &&... \_\_args)
  - template<typename... \_Args>  
[iterator emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool [empty](#) () const noexcept
  - [iterator end](#) () noexcept
  - [local\\_iterator end](#) (size\_type \_\_n)
  - [iterator erase](#) (const\_iterator \_\_first, const\_iterator \_\_last)
  - [allocator\\_type get\\_allocator](#) () const noexcept
  - [hasher hash\\_function](#) () const
  - template<typename \_InputIterator>  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void [insert](#) (initializer\_list< value\_type > \_\_l)
  - [key\\_equal key\\_eq](#) () const
  - float [load\\_factor](#) () const noexcept
  - [size\\_type max\\_bucket\\_count](#) () const noexcept
  - float [max\\_load\\_factor](#) () const noexcept
  - void [max\\_load\\_factor](#) (float \_\_z)
  - [size\\_type max\\_size](#) () const noexcept
  - [unordered\\_multimap & operator=](#) (const [unordered\\_multimap](#) &)=default
  - [unordered\\_multimap & operator=](#) (initializer\_list< value\_type > \_\_l)
  - [unordered\\_multimap & operator=](#) ([unordered\\_multimap](#) &&)=default
  - void [rehash](#) (size\_type \_\_n)
  - void [reserve](#) (size\_type \_\_n)
  - [size\\_type size](#) () const noexcept
  - void [swap](#) ([unordered\\_multimap](#) &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- [const\\_iterator begin](#) () const noexcept
  - [const\\_iterator cbegin](#) () const noexcept
- 
- [const\\_iterator end](#) () const noexcept
  - [const\\_iterator cend](#) () const noexcept
- 
- [iterator insert](#) (const value\_type &\_\_x)
  - [iterator insert](#) (value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair && >::value, iterator > [insert](#) (\_Pair &&\_\_x)
- 
- [iterator insert](#) (const\_iterator \_\_hint, const value\_type &\_\_x)
  - [iterator insert](#) (const\_iterator \_\_hint, value\_type &&\_\_x)
  - template<typename \_Pair>  
\_\_enable\_if\_t< [is\\_constructible](#)< value\_type, \_Pair && >::value, iterator > [insert](#) (const\_iterator \_\_hint, \_Pair &&\_\_x)

- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
  
- `size_type erase (const key_type &__x)`
- `template<__heterogeneous_hash_key< unordered_multimap > _Kt>  
size_type erase (_Kt &&__x)`
  
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
  
  
- `size_type count (const key_type &__x) const`
  
  
- `bool contains (const key_type &__x) const`
- `template<typename _Kt>  
auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)`
  
  
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
  
  
- `size_type bucket (const key_type &__key) const`
- `template<__heterogeneous_hash_key< unordered_multimap > _Kt>  
size_type bucket (const _Kt &__key) const`
  
  
- `const_local_iterator begin (size_type __n) const`
- `const_local_iterator cbegin (size_type __n) const`
  
  
- `const_local_iterator end (size_type __n) const`
- `const_local_iterator cend (size_type __n) const`

## Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>  
bool operator== (const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const  
unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)`



### 6.1027.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
class std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Since

C++11

#### Template Parameters

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Key&gt;</code> .                         |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Key&gt;</code> .                   |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

### 6.1027.2 Member Typedef Documentation

#### allocator\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
allocator_type
Public typedefs.
```

#### const\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
const_iterator
Iterator-related typedefs.
```

#### const\_local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
const_local_iterator
Iterator-related typedefs.
```

#### const\_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::const_pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
Iterator-related typedefs.
```

### const\_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::const_reference
Iterator-related typedefs.
```

### difference\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::difference_type
Iterator-related typedefs.
```

### hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
Public typedefs.
```

### iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
Iterator-related typedefs.
```

### key\_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

### key\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

### local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator
Iterator-related typedefs.
```

**mapped\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::mapped_type
mapped_type
Public typedefs.
```

**pointer**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer
Iterator-related typedefs.
```

**reference**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::reference
reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::size_type
size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::value_type
value_type
Public typedefs.
```

**6.1027.3 Constructor & Destructor Documentation****unordered\_multimap() [1/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_multimap () [default]
Default constructor.
```

**unordered\_multimap() [2/7]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_multimap (
 size_type __n,
 const hasher & __hf = hasher(),
```

```
const key_equal & __eq1 = key_equal(),
const allocator_type & __a = allocator_type() [inline], [explicit]
```

Default constructor creates no elements.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

#### unordered\_multimap() [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type() [inline]
```

Builds an unordered\_multimap from a range.

#### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq1</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first,__last)`).

#### unordered\_multimap() [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

**unordered\_multimap()** [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

**unordered\_multimap()** [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_multimap with no elements.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

**unordered\_multimap()** [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_multimap from an initializer\_list.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_multimap consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

## 6.1027.4 Member Function Documentation

### begin() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

### begin() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered\_multimap.

### begin() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>_↵</code>  | The bucket index. |
| <code>__n</code> |                   |

#### Returns

A read/write local iterator.

### begin() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>_↵</code>  | The bucket index. |
| <code>__n</code> |                   |

#### Returns

A read-only local iterator.

**bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_multimap.

**cbegin() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

**cbegin() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

**cend() [2/2]**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↔</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an `unordered_multimap`. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
 const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                   |                                |
|-------------------|--------------------------------|
| <code>_Key</code> | Key of elements to be located. |
| <code>__x</code>  |                                |

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                   |                                |
|-------------------|--------------------------------|
| <code>_Key</code> | Key of elements to be located. |
| <code>__x</code>  |                                |

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```



```
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

#### Parameters

|                  |               |
|------------------|---------------|
| <code>__x</code> | Key to count. |
|------------------|---------------|

#### Returns

Number of elements with specified key.

#### emplace()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

#### Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered\_multimap. Insertion requires amortized constant time.

#### emplace\_hint()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.

#### Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                |
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the element with key of the `std::pair` built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**empty()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the `unordered_multimap` is empty.

**end()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multimap`.

**end()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_multimap`.

**end()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read/write local iterator.

**end()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

#### Returns

A read-only local iterator.

#### equal\_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair< iterator, iterator > std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::↵
equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

#### equal\_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::pair< const_iterator, const_iterator > std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _↵
Alloc >::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

**erase()** [1/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<__heterogeneous_hash_key< unordered_multimap > _Kt>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 _Kt && __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__x</code> | Key of elements to be erased. |
|------------------|-------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [2/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                               |
|------------------|-------------------------------|
| <code>__x</code> | Key of elements to be erased. |
|------------------|-------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an `unordered_multimap`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [5/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an `unordered_multimap`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_multimap.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**find()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_multimap.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>__x</code>  |                    |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

**get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered\_multimap.

**hash\_function()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_multimap was constructed.

**insert()** [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _InputIterator>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

**insert()** [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair>
__enable_if_t<is_constructible<value_type, _Pair &&>::value, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc>::insert (
 _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::insert (
 const value_type & __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**insert()** [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |



**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

**insert()** [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 initializer_list<value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_multimap`.

**Parameters**

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>__l</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
|------------------|------------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_multimap was constructed.

**load\_factor()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multimap.

**max\_load\_factor()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multimap tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the unordered\_multimap maximum load factor.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

#### max\_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multimap.

#### operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

#### operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_multimap list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_multimap and that the resulting unordered\_multimap's size is the same as the number of elements assigned.

#### operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

**rehash()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_multimap.

**Parameters**

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multimap maximum load factor.

**reserve()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_multimap for a specified number of elements.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

**size()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline],
[noexcept]
```

Returns the size of the unordered\_multimap.

**swap()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
 unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_multimap.

**Parameters**

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>__x</code> | An unordered_multimap of the same element and allocator types. |
|------------------|----------------------------------------------------------------|

This exchanges the elements between two unordered\_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

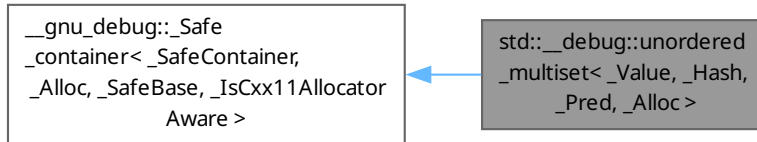
The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 6.1028 std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

Inheritance diagram for std::\_\_debug::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >:



### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, unordered\_multiset > **const\_iterator**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_const\_local\_iterator, unordered\_multiset > **const\_local\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, unordered\_multiset > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_local\_iterator, unordered\_multiset > **local\_iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

### Public Member Functions

- **unordered\_multiset** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (const allocator\_type &\_\_a)

- **unordered\_multiset** (const unordered\_multiset &)=default
- **unordered\_multiset** (const unordered\_multiset &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (unordered\_multiset &&)=default
- **unordered\_multiset** (unordered\_multiset &&\_\_uset, const allocator\_type &\_\_a) noexcept(noexcept(\_↵ Base(std::move(\_\_uset), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local\_iterator **begin** (size\_type \_\_b)
- const\_local\_iterator **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const\_iterator **cbegin** () const noexcept
- const\_local\_iterator **cbegin** (size\_type \_\_b) const
- const\_iterator **cend** () const noexcept
- const\_local\_iterator **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- local\_iterator **end** (size\_type \_\_b)
- const\_local\_iterator **end** (size\_type \_\_b) const
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_it)
- template<\_\_heterogeneous\_hash\_key< unordered\_multiset > \_Kt>  
size\_type **erase** (\_Kt &&\_\_key)
- size\_type **erase** (const key\_type &\_\_key)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_it)
- iterator **erase** (iterator \_\_it)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, value\_type &&\_\_obj)

- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (value\_type &&\_\_obj)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- unordered\_multiset & **operator=** (const unordered\_multiset &)=default
- unordered\_multiset & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- unordered\_multiset & **operator=** (unordered\_multiset &&)=default
- void **swap** (unordered\_multiset &\_\_x) noexcept([noexcept](#)([declval](#)< \_Base & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
class `::__gnu_debug::Safe_iterator`
- `template<typename _ItT, typename _SeqT>`  
class `::__gnu_debug::Safe_local_iterator`

#### 6.1028.1 Detailed Description

`template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>,`  
`typename _Alloc = std::allocator<_Value>>`  
`class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`

Class std::unordered\_multiset with safety/checking/debug instrumentation.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 6.1029 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- `unordered_multiset ()`=default
- `template<typename _InputIterator>`  
`unordered_multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)`
- `template<typename _InputIterator>`  
`unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)`
- `template<typename _InputIterator>`  
`unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)`
- `template<typename _InputIterator>`  
`unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_multiset (const allocator_type &__a)`
- `unordered_multiset (const unordered_multiset &)=default`
- `unordered_multiset (const unordered_multiset &__umset, const allocator_type &__a)`
- `unordered_multiset (initializer_list< value_type > __l, const allocator_type &__a)`
- `unordered_multiset (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)`
- `unordered_multiset (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)`
- `unordered_multiset (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_multiset (size_type __n, const allocator_type &__a)`
- `unordered_multiset (size_type __n, const hasher &__hf, const allocator_type &__a)`
- `unordered_multiset (size_type __n, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`
- `unordered_multiset (unordered_multiset &&)=default`
- `unordered_multiset (unordered_multiset &&__umset, const allocator_type &__a) noexcept(noexcept(Hashtable(std::move(__umset._M_h), __a)))`
- `size_type bucket_count ()` const noexcept
- `size_type bucket_size (size_type __n)` const
- `const_iterator cbegin ()` const noexcept
- `const_iterator cend ()` const noexcept
- `void clear ()` noexcept
- `template<typename... _Args>`  
`iterator emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __pos, _Args &&... __args)`
- `bool empty ()` const noexcept
- `template<__heterogeneous_hash_key< unordered_multiset > _Kt>`  
`size_type erase (_Kt &&__x)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `allocator_type get_allocator ()` const noexcept
- `hasher hash_function ()` const
- `template<typename _InputIterator>`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `key_equal key_eq ()` const
- `float load_factor ()` const noexcept
- `size_type max_bucket_count ()` const noexcept
- `float max_load_factor ()` const noexcept



- void `max_load_factor` (float \_\_z)
  - `size_type max_size` () const noexcept
  - `unordered_multiset` & `operator=` (const `unordered_multiset` &)=default
  - `unordered_multiset` & `operator=` (initializer\_list< value\_type > \_\_l)
  - `unordered_multiset` & `operator=` (`unordered_multiset` &&)=default
  - void `rehash` (size\_type \_\_n)
  - void `reserve` (size\_type \_\_n)
  - `size_type size` () const noexcept
  - void `swap` (`unordered_multiset` &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- `iterator begin` () noexcept
  - `const_iterator begin` () const noexcept
- 
- `iterator end` () noexcept
  - `const_iterator end` () const noexcept
- 
- `iterator insert` (const value\_type &\_\_x)
  - `iterator insert` (value\_type &&\_\_x)
- 
- `iterator insert` (const\_iterator \_\_hint, const value\_type &\_\_x)
  - `iterator insert` (const\_iterator \_\_hint, value\_type &&\_\_x)
- 
- `iterator erase` (const\_iterator \_\_position)
  - `iterator erase` (iterator \_\_position)
- 
- `iterator find` (const key\_type &\_\_x)
  - `const_iterator find` (const key\_type &\_\_x) const
- 
- `size_type count` (const key\_type &\_\_x) const
- 
- bool `contains` (const key\_type &\_\_x) const
  - template<typename \_Kt>  
auto `contains` (const \_Kt &\_\_x) const -> decltype(\_M\_h.\_M\_find\_tr(\_\_x), void(), true)
- 
- std::pair< iterator, iterator > `equal_range` (const key\_type &\_\_x)
  - std::pair< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const
- 
- `size_type bucket` (const key\_type &\_\_key) const

- `template<__heterogeneous_hash_key< unordered_multiset > _Kt>  
size_type bucket (const _Kt &__key) const`
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `const_local_iterator cbegin (size_type __n) const`
- `local_iterator end (size_type __n)`
- `const_local_iterator end (size_type __n) const`
- `const_local_iterator cend (size_type __n) const`

## Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>  
bool operator== (const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset<  
_Value1, _Hash1, _Pred1, _Alloc1 > &)`

## 6.1029.1 Detailed Description

`template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, type-  
name _Alloc = allocator<_Value>>  
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Since

C++11

## Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)  
Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

## 6.1029.2 Member Typedef Documentation

### `allocator_type`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↵
allocator_type
Public typedefs.
```

**const\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::const_iterator
Iterator-related typedefs.
```

**const\_local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::const_local_iterator
Iterator-related typedefs.
```

**const\_pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::const_pointer
Iterator-related typedefs.
```

**const\_reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::const_reference
Iterator-related typedefs.
```

**difference\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::difference_type
Iterator-related typedefs.
```

**hasher**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::hasher
Public typedefs.
```

**iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::iterator
Iterator-related typedefs.
```

**key\_equal**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

**key\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

**local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::local_iterator
Iterator-related typedefs.
```

**pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
Iterator-related typedefs.
```

**reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type
Public typedefs.
```

**6.1029.3 Constructor & Destructor Documentation****unordered\_multiset()** [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
```

```

typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset () [default]
Default constructor.

```

**unordered\_multiset()** [2/7]

```

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]

```

Default constructor creates no elements.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

**unordered\_multiset()** [3/7]

```

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]

```

Builds an unordered\_multiset from a range.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq1</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_multiset` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

#### `unordered_multiset()` [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

#### `unordered_multiset()` [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

#### `unordered_multiset()` [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_multiset` from an `initializer_list`.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An <code>initializer_list</code> . |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an `unordered_multiset` consisting of copies of the elements in the list. This is linear in `N` (where `N` is `__l.size()`).

#### `unordered_multiset()` [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_multiset with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

### 6.1029.4 Member Function Documentation

#### begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

#### begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

#### begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

#### begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_multiset.

**cbegin() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

**cbegin() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.



**cend()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                   |                   |
|-------------------|-------------------|
| $\leftrightarrow$ | The bucket index. |
| $n$               |                   |

**Returns**

A read-only local iterator.

**clear()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**contains()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
 const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void()), true) [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                   |                                |
|-------------------|--------------------------------|
| $\leftrightarrow$ | Key of elements to be located. |
| $x$               |                                |

**Returns**

True if there is any element with the specified key.

**contains()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <code>_↔</code> | Element to located. |
| <code>_X</code> |                     |

**Returns**

Number of elements with specified key.

**emplace()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Builds and insert an element into the unordered\_multiset.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
```

```
const_iterator __pos,
_Args &&... __args) [inline]
```

Inserts an element into the unordered\_multiset.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

#### Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant time.

#### empty()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered\_multiset is empty.

#### end() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

#### end() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

#### end() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                        |                   |
|------------------------|-------------------|
| <code>↵<br/>__n</code> | The bucket index. |
|------------------------|-------------------|

**Returns**

A read-only local iterator.

**end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

**equal\_range()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_
_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**equal\_range()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__key</code> | Key to be located. |
| <code>__x</code>   |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

**erase() [1/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__key</code> | Key of element to be erased. |
| <code>__x</code>   |                              |

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [2/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_multiset.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an unordered\_multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**find()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_multiset.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

**get\_allocator()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered\_multiset.

**hash\_function()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_multiset was constructed.

**insert()** [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that inserts a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

#### `insert()` [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts an element into the `unordered_multiset`.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

#### Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

#### `insert()` [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Inserts an element into the `unordered_multiset`.

#### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

#### Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.



**insert()** [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Inserts a list of elements into the unordered\_multiset.

**Parameters**

|                                                                                            |                                                                 |
|--------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| <code>↵</code><br><code>__↵</code><br><code>↵</code><br><code>__↵</code><br><code>/</code> | A std::initializer_list<value_type> of elements to be inserted. |
|--------------------------------------------------------------------------------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

**insert()** [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                 |                         |
|-----------------|-------------------------|
| <code>_↔</code> | Element to be inserted. |
| <code>_X</code> |                         |

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the `unordered_multiset` was constructed.

**load\_factor()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the `unordered_multiset`.

**max\_load\_factor() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the `unordered_multiset` tries to keep the load factor less than or equal to.

**max\_load\_factor() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the `unordered_multiset` maximum load factor.

**Parameters**

|                 |                              |
|-----------------|------------------------------|
| <code>_↔</code> | The new maximum load factor. |
| <code>_Z</code> |                              |

**max\_size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multiset.

**operator=()** [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

**operator=()** [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_multiset list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills an unordered\_multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_multiset and that the resulting unordered\_multiset's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

**rehash()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_multiset.

**Parameters**

|                                      |                            |
|--------------------------------------|----------------------------|
| $\leftrightarrow$<br><code>_n</code> | The new number of buckets. |
|--------------------------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the `unordered_multiset` maximum load factor.

**reserve()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the `unordered_multiset` for a specified number of elements.

**Parameters**

|                                      |                              |
|--------------------------------------|------------------------------|
| $\leftrightarrow$<br><code>_n</code> | Number of elements required. |
|--------------------------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

**size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the `unordered_multiset`.

**swap()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another `unordered_multiset`.

**Parameters**

|                                      |                                                                             |
|--------------------------------------|-----------------------------------------------------------------------------|
| $\leftrightarrow$<br><code>_X</code> | An <code>unordered_multiset</code> of the same element and allocator types. |
|--------------------------------------|-----------------------------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

**6.1030 std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference**

```
#include <unordered_set>
```

Inheritance diagram for std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >:



## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_const\_iterator, unordered\_set > **const\_iterator**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_const\_local\_iterator, unordered\_set > **const\_local\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_Base::hasher **hasher**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_iterator](#)< \_Base\_iterator, unordered\_set > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef [\\_\\_gnu\\_debug::\\_Safe\\_local\\_iterator](#)< \_Base\_local\_iterator, unordered\_set > **local\_iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_set** (\_Base\_ref \_\_x)
- template<typename \_InputIterator>  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator>  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const allocator\_type &\_\_a)
- **unordered\_set** (const unordered\_set &)=default
- **unordered\_set** (const unordered\_set &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a)
- **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)

- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher & \_\_hf=hasher(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- **unordered\_set** (size\_type \_\_n, const allocator\_type & \_\_a)
- **unordered\_set** (size\_type \_\_n, const hasher & \_\_hf, const allocator\_type & \_\_a)
- **unordered\_set** (size\_type \_\_n, const hasher & \_\_hf=hasher(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- **unordered\_set** (unordered\_set &&)=default
- **unordered\_set** (unordered\_set && \_\_uset, const allocator\_type & \_\_a) noexcept(noexcept(\_Base([std::move](#)(\_\_uset), \_\_a)))
- const \_Base & **\_M\_base** () const noexcept
- \_Base & **\_M\_base** () noexcept
- const\_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local\_iterator **begin** (size\_type \_\_b)
- const\_local\_iterator **begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const\_iterator **cbegin** () const noexcept
- const\_local\_iterator **cbegin** (size\_type \_\_b) const
- const\_iterator **cend** () const noexcept
- const\_local\_iterator **cend** (size\_type \_\_b) const
- void **clear** () noexcept
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- const\_iterator **end** () const noexcept
- iterator **end** () noexcept
- local\_iterator **end** (size\_type \_\_b)
- const\_local\_iterator **end** (size\_type \_\_b) const
- [std::pair](#)< iterator, iterator > **equal\_range** (const key\_type & \_\_key)
- [std::pair](#)< const\_iterator, const\_iterator > **equal\_range** (const key\_type & \_\_key) const
- \_Base\_iterator **erase** (\_Base\_const\_iterator \_\_it)
- template<\_\_heterogeneous\_hash\_key< unordered\_set > \_Kt>  
size\_type **erase** (\_Kt && \_\_key)
- size\_type **erase** (const key\_type & \_\_key)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **erase** (const\_iterator \_\_it)
- iterator **erase** (iterator \_\_it)
- iterator **find** (const key\_type & \_\_key)
- const\_iterator **find** (const key\_type & \_\_key) const
- template<typename \_InputIterator>  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<\_\_heterogeneous\_hash\_key< unordered\_set > \_Kt>  
[std::pair](#)< iterator, bool > **insert** (\_Kt && \_\_obj)
- [std::pair](#)< iterator, bool > **insert** (const value\_type & \_\_obj)
- template<\_\_heterogeneous\_hash\_key< unordered\_set > \_Kt>  
iterator **insert** (const\_iterator \_\_hint, \_Kt && \_\_obj)
- iterator **insert** (const\_iterator \_\_hint, const value\_type & \_\_obj)
- iterator **insert** (const\_iterator \_\_hint, value\_type && \_\_obj)
- void **insert** ([std::initializer\\_list](#)< value\_type > \_\_l)
- [std::pair](#)< iterator, bool > **insert** (value\_type && \_\_obj)

- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- unordered\_set & **operator=** (const unordered\_set &)=default
- unordered\_set & **operator=** (initializer\_list< value\_type > \_\_l)
- unordered\_set & **operator=** (unordered\_set &&)=default
- void **swap** (unordered\_set &\_\_x) noexcept(noexcept([declval](#)< \_Base & >().swap(\_\_x)))

### Protected Member Functions

- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept

### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::[\\_\\_gnu\\_debug::Safe\\_iterator](#)
- template<typename \_ItT, typename \_SeqT>  
class ::[\\_\\_gnu\\_debug::Safe\\_local\\_iterator](#)

### 6.1030.1 Detailed Description

**template<typename \_Value, typename \_Hash = std::hash<\_Value>, typename \_Pred = std::equal\_to<\_Value>,  
typename \_Alloc = std::allocator<\_Value>>>**  
**class std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >**

Class std::unordered\_set with safety/checking/debug instrumentation.  
 The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

## 6.1031 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

```
#include <unordered_set>
```

### Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
- typedef \_Hashtable::value\_type [value\\_type](#)
- typedef \_Hashtable::hasher [hasher](#)
- typedef \_Hashtable::key\_equal [key\\_equal](#)
- typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- typedef \_Hashtable::pointer [pointer](#)
- typedef \_Hashtable::const\_pointer [const\\_pointer](#)
- typedef \_Hashtable::reference [reference](#)
- typedef \_Hashtable::const\_reference [const\\_reference](#)
- typedef \_Hashtable::iterator [iterator](#)
- typedef \_Hashtable::const\_iterator [const\\_iterator](#)
- typedef \_Hashtable::local\_iterator [local\\_iterator](#)
- typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
- typedef \_Hashtable::size\_type [size\\_type](#)
- typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- `unordered_set` ()=default
- `template<typename _InputIterator>`  
`unordered_set` (\_InputIterator \_\_first, \_InputIterator \_\_last, const `allocator_type` &\_\_a)
- `template<typename _InputIterator>`  
`unordered_set` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `template<typename _InputIterator>`  
`unordered_set` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `template<typename _InputIterator>`  
`unordered_set` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_set` (const `allocator_type` &\_\_a)
- `unordered_set` (const `unordered_set` &)=default
- `unordered_set` (const `unordered_set` &\_\_uset, const `allocator_type` &\_\_a)
- `unordered_set` (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- `unordered_set` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_set` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_set` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_set` (`size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_set` (`size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_set` (`size_type` \_\_n, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_set` (`unordered_set` &&)=default
- `unordered_set` (`unordered_set` &&\_\_uset, const `allocator_type` &\_\_a) noexcept(noexcept(\_Hashtable(`std::move`(↔ \_\_uset.\_M\_h), \_\_a)))
- `size_type` `bucket_count` () const noexcept
- `size_type` `bucket_size` (`size_type` \_\_n) const
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `cend` () const noexcept
- `void` `clear` () noexcept
- `template<typename... _Args>`  
`std::pair`< `iterator`, bool > `emplace` (\_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator` `emplace_hint` (const `iterator` \_\_pos, \_Args &&... \_\_args)
- `bool` `empty` () const noexcept
- `template<__heterogeneous_hash_key< unordered_set > _Kt>`  
`size_type` `erase` (\_Kt &&\_\_x)
- `size_type` `erase` (const `key_type` &\_\_x)
- `iterator` `erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type` `get_allocator` () const noexcept
- `hasher` `hash_function` () const
- `template<typename _InputIterator>`  
`void` `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `void` `insert` (`initializer_list`< `value_type` > \_\_l)
- `key_equal` `key_eq` () const
- `float` `load_factor` () const noexcept
- `size_type` `max_bucket_count` () const noexcept
- `float` `max_load_factor` () const noexcept
- `void` `max_load_factor` (float \_\_z)



- `size_type max_size ()` const noexcept
  - `unordered_set & operator= (const unordered_set &)=default`
  - `unordered_set & operator= (initializer_list< value_type > __l)`
  - `unordered_set & operator= (unordered_set &&)=default`
  - `void rehash (size_type __n)`
  - `void reserve (size_type __n)`
  - `size_type size ()` const noexcept
  - `void swap (unordered_set &__x)` noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- `iterator begin ()` noexcept
  - `const_iterator begin ()` const noexcept
- 
- `iterator end ()` noexcept
  - `const_iterator end ()` const noexcept
- 
- `std::pair< iterator, bool > insert (const value_type &__x)`
  - `std::pair< iterator, bool > insert (value_type &&__x)`
  - `template<__heterogeneous_hash_key< unordered_set > _Kt>  
std::pair< iterator, bool > insert (_Kt &&__x)`
- 
- `iterator insert (const_iterator __hint, const value_type &__x)`
  - `iterator insert (const_iterator __hint, value_type &&__x)`
  - `template<__heterogeneous_hash_key< unordered_set > _Kt>  
iterator insert (const_iterator, _Kt &&__x)`
- 
- `iterator erase (const_iterator __position)`
  - `iterator erase (iterator __position)`
- 
- `iterator find (const key_type &__x)`
  - `const_iterator find (const key_type &__x)` const
- 
- `size_type count (const key_type &__x)` const
- 
- `bool contains (const key_type &__x)` const
  - `template<typename _Kt>  
auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)`
- 
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
  - `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x)` const

- `size_type bucket` (const `key_type` & \_\_key) const
- `template<__heterogeneous_hash_key< unordered_set > _Kt>`  
`size_type bucket` (const \_Kt & \_\_key) const
- `local_iterator begin` (size\_type \_\_n)
- `const_local_iterator begin` (size\_type \_\_n) const
- `const_local_iterator cbegin` (size\_type \_\_n) const
- `local_iterator end` (size\_type \_\_n)
- `const_local_iterator end` (size\_type \_\_n) const
- `const_local_iterator cend` (size\_type \_\_n) const

## Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>`  
`bool operator==` (const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &)

### 6.1031.1 Detailed Description

`template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>`  
`class std::unordered_set< _Value, _Hash, _Pred, _Alloc >`

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Since

C++11

## Template Parameters

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

### 6.1031.2 Member Typedef Documentation

#### `allocator_type`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator_type
Public typedefs.
```

**const\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_↵
iterator
```

Iterator-related typedefs.

**const\_local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::↵
const_local_iterator
```

Iterator-related typedefs.

**const\_pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_↵
pointer
```

Iterator-related typedefs.

**const\_reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_↵
reference
```

Iterator-related typedefs.

**difference\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::difference↵
_type
```

Iterator-related typedefs.

**hasher**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_set<_Value, _Hash, _Pred, _Alloc>::hasher
```

Public typedefs.

**iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::iterator
```

Iterator-related typedefs.

**key\_equal**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_equal
Public typedefs.
```

**key\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_type
Public typedefs.
```

**local\_iterator**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::local_↵
iterator
Iterator-related typedefs.
```

**pointer**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer
Iterator-related typedefs.
```

**reference**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference
Iterator-related typedefs.
```

**size\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type
Iterator-related typedefs.
```

**value\_type**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type
Public typedefs.
```

**6.1031.3 Constructor & Destructor Documentation****unordered\_set()** [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set () [default]
```

Default constructor.

### unordered\_set() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

### unordered\_set() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
template<typename _InputIterator>
```

```
std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_set from a range.

#### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_set` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

#### `unordered_set()` [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 const unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

#### `unordered_set()` [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

#### `unordered_set()` [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 const allocator_type & __a) [inline], [explicit]
```

Creates an `unordered_set` with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

#### `unordered_set()` [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_set` from an `initializer_list`.

#### Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> . |
| <code>__n</code> | Minimal initial number of buckets. |

|                    |                         |
|--------------------|-------------------------|
| <code>__hf</code>  | A hash functor.         |
| <code>__eqf</code> | A key equality functor. |
| <code>__a</code>   | An allocator object.    |

Create an `unordered_set` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `l.size()`).

#### 6.1031.4 Member Function Documentation

##### **begin()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

##### **begin()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

##### **begin()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

##### **begin()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                      |                   |
|----------------------|-------------------|
| $\_ \leftrightarrow$ | The bucket index. |
| $\_n$                |                   |

**Returns**

A read-only local iterator.

**bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_set.

**cbegin() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

**cbegin() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                      |                   |
|----------------------|-------------------|
| $\_ \leftrightarrow$ | The bucket index. |
| $\_n$                |                   |

**Returns**

A read-only local iterator.

**cend() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

**cend() [2/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```



```
const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

#### Returns

A read-only local iterator.

#### clear()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### contains() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set<_Value, _Hash, _Pred, _Alloc>::contains (
 const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true) [inline]
```

Finds whether an element with the given key exists.

#### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

#### Returns

True if there is any element with the specified key.

#### contains() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set<_Value, _Hash, _Pred, _Alloc>::contains (
 const key_type & __x) const [inline]
```

Finds whether an element with the given key exists.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <code>_↔</code> | Key of elements to be located. |
| <code>_X</code> |                                |

**Returns**

True if there is any element with the specified key.

**count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                 |                     |
|-----------------|---------------------|
| <code>_↔</code> | Element to located. |
| <code>_X</code> |                     |

**Returns**

Number of elements with specified key.

This function only makes sense for `unordered_multisets`; for `unordered_set` the result will either be 0 (not present) or 1 (present).

**emplace()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert an element into the `unordered_set`.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to build and insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

**emplace\_hint()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant time.

**empty()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered\_set is empty.

**end() [1/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

**end() [2/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_set.

**end() [3/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                        |                   |
|------------------------|-------------------|
| <code>_↔<br/>_n</code> | The bucket index. |
|------------------------|-------------------|

#### Returns

A read-only local iterator.

#### **end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

#### Parameters

|                        |                   |
|------------------------|-------------------|
| <code>_↔<br/>_n</code> | The bucket index. |
|------------------------|-------------------|

#### Returns

A read-only local iterator.

#### **equal\_range()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range
(
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                        |                    |
|------------------------|--------------------|
| <code>_↔<br/>_x</code> | Key to be located. |
|------------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

#### **equal\_range()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::pair< const_iterator, const_iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

#### erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

#### Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered\_set. For an unordered\_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an unordered\_set.

#### Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [3/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an `unordered_set`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**erase() [4/4]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an `unordered_set`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

**find() [1/2]**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

#### find() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an unordered\_set.

#### Parameters

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

#### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( end() ) iterator.

#### get\_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Returns the allocator object used by the unordered\_set.

#### hash\_function()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set<_Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_set was constructed.

#### insert() [1/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
template<typename _InputIterator>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

#### insert() [2/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<__heterogeneous_hash_key< unordered_set > _Kt>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 _Kt && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

#### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`. Insertion requires amortized constant time.

#### insert() [3/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|



**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

**insert()** [4/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [5/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [6/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<__heterogeneous_hash_key< unordered_set > _Kt>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator ,
 _Kt && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant.

**insert()** [7/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the `unordered_set`.

**Parameters**

|                |                                                                                    |
|----------------|------------------------------------------------------------------------------------|
| ↵              | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| ↵              |                                                                                    |
| ↵              |                                                                                    |
| ↵              |                                                                                    |
| <code>l</code> |                                                                                    |

Complexity similar to that of the range constructor.

**insert()** [8/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Attempts to insert an element into the unordered\_set.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

**key\_eq()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_set was constructed.

**load\_factor()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline], [noexcept]
```

Returns the average number of elements per bucket.

**max\_bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered\_set.

**max\_load\_factor()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_set tries to keep the load factor less than or equal to.

**max\_load\_factor()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the unordered\_set maximum load factor.

#### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__z</code> | The new maximum load factor. |
|------------------|------------------------------|

#### max\_size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the unordered\_set.

#### operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 const unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

#### operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_set list assignment operator.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills an unordered\_set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned.

#### operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

**rehash()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_set.

**Parameters**

|                     |                            |
|---------------------|----------------------------|
| $\leftarrow$<br>__n | The new number of buckets. |
|---------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

**reserve()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_set for a specified number of elements.

**Parameters**

|                     |                              |
|---------------------|------------------------------|
| $\leftarrow$<br>__n | Number of elements required. |
|---------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

**size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the unordered\_set.

**swap()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
 unordered_set< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_set.

**Parameters**

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| $\leftarrow$<br>__x | An unordered_set of the same element and allocator types. |
|---------------------|-----------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 6.1032 std::pmr::unsynchronized\_pool\_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::unsynchronized\_pool\_resource:



### Public Member Functions

- **unsynchronized\_pool\_resource** (const [pool\\_options](#) &\_\_opts)
- **unsynchronized\_pool\_resource** (const [pool\\_options](#) &\_\_opts, memory\_resource \* \_\_upstream)
- **unsynchronized\_pool\_resource** (const unsynchronized\_pool\_resource &)=delete
- **unsynchronized\_pool\_resource** (memory\_resource \* \_\_upstream)
- void \* **allocate** (size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- void **deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment=\_S\_max\_align)
- bool **is\_equal** (const memory\_resource & \_\_other) const noexcept
- unsynchronized\_pool\_resource & **operator=** (const unsynchronized\_pool\_resource &)=delete
- [pool\\_options](#) **options** () const noexcept
- void **release** ()
- memory\_resource \* **upstream\_resource** () const noexcept

### Protected Member Functions

- void \* **do\_allocate** (size\_t \_\_bytes, size\_t \_\_alignment) override
- void **do\_deallocate** (void \* \_\_p, size\_t \_\_bytes, size\_t \_\_alignment) override
- bool **do\_is\_equal** (const memory\_resource & \_\_other) const noexcept override

#### 6.1032.1 Detailed Description

A non-thread-safe memory resource that manages pools of fixed-size blocks.

Since

C++17

**6.1032.2 Member Function Documentation****do\_allocate()**

```
void * std::pmr::unsynchronized_pool_resource::do_allocate (
 size_t __bytes,
 size_t __alignment) [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

**do\_deallocate()**

```
void std::pmr::unsynchronized_pool_resource::do_deallocate (
 void * __p,
 size_t __bytes,
 size_t __alignment) [override], [protected], [virtual]
```

Implements [std::pmr::memory\\_resource](#).

**do\_is\_equal()**

```
bool std::pmr::unsynchronized_pool_resource::do_is_equal (
 const memory_resource & __other) const [inline], [override], [protected], [virtual],
[noexcept]
```

Implements [std::pmr::memory\\_resource](#).

The documentation for this class was generated from the following file:

- [memory\\_resource](#)

**6.1033 `std::uses_allocator< typename, typename >` Struct Template Reference****6.1033.1 Detailed Description**

**template<typename, typename>**

**struct std::uses\_allocator< typename, typename >**

Declare `uses_allocator` so it can be specialized in `<queue>` etc.

The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

**6.1034 `std::uses_allocator< tuple< _Types... >, _Alloc >` Struct Template Reference**

```
#include <tuple>
```

**6.1034.1 Detailed Description**

**template<typename... \_Types, typename \_Alloc>**

**struct std::uses\_allocator< tuple< \_Types... >, \_Alloc >**

Partial specialization for tuples.

The documentation for this struct was generated from the following file:

- [tuple](#)

**6.1035 `std::chrono::utc_clock` Class Reference**

```
#include <chrono>
```

### Public Types

- using **duration**
- using **period**
- using **rep**
- using **time\_point**

### Static Public Member Functions

- `template<typename _Duration>`  
`static utc_time< common\_type\_t< _Duration, seconds > > from_sys (const sys_time< _Duration > &__t)`
- `static time_point now ()`
- `template<typename _Duration>`  
`static sys_time< common\_type\_t< _Duration, seconds > > to_sys (const utc_time< _Duration > &__t)`

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 6.1035.1 Detailed Description

A clock that measures Universal Coordinated Time (UTC).

The epoch is 1970-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- [chrono](#)

### 6.1036 `std::valarray<_Tp>` Class Template Reference

```
#include <valarray>
```

#### Public Types

- `typedef const _Tp * const_iterator`
- `typedef _Tp * iterator`
- `typedef _Tp value_type`

#### Public Member Functions

- [valarray](#) () noexcept
- `template<class _Dom>`  
`valarray (const _Expr< _Dom, _Tp > &__e)`
- [valarray](#) (const \_Tp &, size\_t)
- `template<typename _Tp>`  
`valarray (const _Tp *__restrict __p, size_t __n)`
- [valarray](#) (const \_Tp \*\_\_restrict \_\_p, size\_t)
- [valarray](#) (const [gslice\\_array](#)< \_Tp > &)
- [valarray](#) (const [indirect\\_array](#)< \_Tp > &)
- [valarray](#) (const [mask\\_array](#)< \_Tp > &)
- [valarray](#) (const [slice\\_array](#)< \_Tp > &)
- [valarray](#) (const valarray &)



- [valarray](#) ([initializer\\_list](#)<\_Tp>)
- [valarray](#) (size\_t)
- [valarray](#) (valarray &&) noexcept
- [\\_Expr](#)<\_ValFunClos<\_ValArray, \_Tp>, \_Tp> [apply](#) (\_Tp \_\_func(\_Tp)) const
- [\\_Expr](#)<\_RefFunClos<\_ValArray, \_Tp>, \_Tp> [apply](#) (\_Tp \_\_func(const \_Tp &)) const
- [const\\_iterator](#) **begin** () const noexcept
- [iterator](#) **begin** () noexcept
- [valarray](#)<\_Tp> [cshift](#) (int \_\_n) const
- [const\\_iterator](#) **end** () const noexcept
- [iterator](#) **end** () noexcept
- [\\_Tp](#) **max** () const
- [\\_Tp](#) **min** () const
- [\\_UnaryOp](#)<\_\_logical\_not>::\_Rt [operator!](#) () const
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator%=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator%=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator%=** (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator&=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator&=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator&=** (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator\*=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator\*=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator\*=** (const [valarray](#)<\_Tp> &)
- [\\_UnaryOp](#)<\_\_unary\_plus>::\_Rt **operator+** () const
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator+=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator+=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator+=** (const [valarray](#)<\_Tp> &)
- [\\_UnaryOp](#)<\_\_negate>::\_Rt **operator-** () const
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator-=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator-=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator-=** (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator/=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator/=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator/=** (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator<<=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator<<=** (const \_Tp &)
- [valarray](#)<\_Tp> & **operator<<=** (const [valarray](#)<\_Tp> &)
- template<class \_Dom>  
  [valarray](#)<\_Tp> & **operator=** (const [\\_Expr](#)<\_Dom, \_Tp> &)
- [valarray](#)<\_Tp> & **operator=** (const \_Tp &\_\_t)
- [valarray](#)<\_Tp> & **operator=** (const [gslice\\_array](#)<\_Tp> &\_\_ga)
- [valarray](#)<\_Tp> & **operator=** (const [indirect\\_array](#)<\_Tp> &\_\_ia)
- [valarray](#)<\_Tp> & **operator=** (const [mask\\_array](#)<\_Tp> &\_\_ma)
- [valarray](#)<\_Tp> & **operator=** (const [slice\\_array](#)<\_Tp> &\_\_sa)
- [valarray](#)<\_Tp> & **operator=** (const [valarray](#)<\_Tp> &\_\_v)

- `valarray & operator= (initializer_list< _Tp > __l)`
- `valarray< _Tp > & operator= (valarray< _Tp > &&__v) noexcept`
- `template<class _Dom>`  
`valarray< _Tp > & operator>= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator>>= (const _Tp &)`
- `valarray< _Tp > & operator>>= (const valarray< _Tp > &)`
- `gslice_array< _Tp > operator[] (const gslice &__s)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > operator[] (const gslice &__s) const`
- `mask_array< _Tp > operator[] (const valarray< bool > &__m)`
- `valarray< _Tp > operator[] (const valarray< bool > &__m) const`
- `indirect_array< _Tp > operator[] (const valarray< size_t > &__i)`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > operator[] (const valarray< size_t > &__i) const`
- `_Tp & operator[] (size_t __i) noexcept`
- `const _Tp & operator[] (size_t) const noexcept`
- `slice_array< _Tp > operator[] (slice __s)`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > operator[] (slice __s) const`
- `template<class _Dom>`  
`valarray< _Tp > & operator^= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator^= (const _Tp &)`
- `valarray< _Tp > & operator^= (const valarray< _Tp > &)`
- `template<class _Dom>`  
`valarray< _Tp > & operator|= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & operator|= (const _Tp &)`
- `valarray< _Tp > & operator|= (const valarray< _Tp > &)`
- `_UnaryOp< __bitwise_not >::Rt operator~ () const`
- `void resize (size_t __size, _Tp __c= _Tp())`
- `valarray< _Tp > shift (int __n) const`
- `size_t size () const`
- `_Tp sum () const`
- `void swap (valarray< _Tp > &__v) noexcept`

## Friends

- `struct _Array< _Tp >`

## 6.1036.1 Detailed Description

**template<class \_Tp>**

**class std::valarray< \_Tp >**

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

## Template Parameters

|                  |                              |
|------------------|------------------------------|
| <code>_Tp</code> | Type of object in the array. |
|------------------|------------------------------|

### 6.1036.2 Constructor & Destructor Documentation

#### valarray()

```
template<class _Tp>
std::valarray<_Tp>::valarray (
 const _Tp * __restrict__,
 size_t)
```

Construct an array initialized to the first  $n$  elements of  $t$ .

References [valarray\(\)](#).

### 6.1036.3 Member Function Documentation

#### begin()

```
template<class _Tp>
iterator std::valarray<_Tp>::begin () [inline], [nodiscard], [noexcept]
```

Return an iterator pointing to the first element of the valarray.

#### end()

```
template<class _Tp>
iterator std::valarray<_Tp>::end () [inline], [nodiscard], [noexcept]
```

Return an iterator pointing to one past the last element of the valarray.

#### operator"!"()

```
template<class _Tp>
_UnaryOp<__logical_not>::_Rt std::valarray<_Tp>::operator! () const
```

Return a new valarray by applying unary ! to each element.

#### operator%=() [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator%= (
 const _Tp &)
```

Set each element  $e$  of array to  $e \% t$ .

References [valarray\(\)](#).

#### operator%=() [2/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator%= (
 const valarray<_Tp> &)
```

Modulo elements of array by corresponding elements of  $v$ .

References [valarray\(\)](#).

#### operator&=() [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator&= (
 const _Tp &)
```

Set each element  $e$  of array to  $e \& t$ .

References [valarray\(\)](#).

**operator&=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
 const valarray< _Tp > &)
```

Logical and corresponding elements of *v* with elements of array.

References [valarray\(\)](#).

**operator\*=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
 const _Tp &)
```

Multiply each element of array by *t*.

References [valarray\(\)](#).

**operator\*=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
 const valarray< _Tp > &)
```

Multiply elements of array by corresponding elements of *v*.

References [valarray\(\)](#).

**operator+()**

```
template<class _Tp>
UnaryOp< __unary_plus >::Rt std::valarray< _Tp >::operator+ () const
```

Return a new valarray by applying unary + to each element.

**operator+=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
 const _Tp &)
```

Add *t* to each element of array.

References [valarray\(\)](#).

**operator+=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
 const valarray< _Tp > &)
```

Add corresponding elements of *v* to elements of array.

References [valarray\(\)](#).

**operator-()**

```
template<class _Tp>
UnaryOp< __negate >::Rt std::valarray< _Tp >::operator- () const
```

Return a new valarray by applying unary - to each element.

**operator-=()** [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator-= (
 const _Tp &)
```

Subtract *t* to each element of array.

References [valarray\(\)](#).

**operator-=()** [2/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator-= (
 const valarray<_Tp> &)
```

Subtract corresponding elements of *v* from elements of array.

References [valarray\(\)](#).

**operator/=()** [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator/= (
 const _Tp &)
```

Divide each element of array by *t*.

References [valarray\(\)](#).

**operator/=()** [2/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator/= (
 const valarray<_Tp> &)
```

Divide elements of array by corresponding elements of *v*.

References [valarray\(\)](#).

**operator<<=()** [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator<<= (
 const _Tp &)
```

Left shift each element *e* of array by *t* bits.

References [valarray\(\)](#).

**operator<<=()** [2/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator<<= (
 const valarray<_Tp> &)
```

Left shift elements of array by corresponding elements of *v*.

**operator>>=()** [1/2]

```
template<class _Tp>
valarray<_Tp> & std::valarray<_Tp>::operator>>= (
 const _Tp &)
```

Right shift each element *e* of array by *t* bits.

References [valarray\(\)](#).

**operator>>=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
 const valarray< _Tp > &)
```

Right shift elements of array by corresponding elements of *v*.

References [valarray\(\)](#).

**operator^=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
 const _Tp &)
```

Set each element *e* of array to  $e \wedge t$ .

References [valarray\(\)](#).

**operator^=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
 const valarray< _Tp > &)
```

Logical xor corresponding elements of *v* with elements of array.

References [valarray\(\)](#).

**operator|=()** [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
 const _Tp &)
```

Set each element *e* of array to  $e \mid t$ .

**operator|=()** [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
 const valarray< _Tp > &)
```

Logical or corresponding elements of *v* with elements of array.

References [valarray\(\)](#).

**operator~()**

```
template<class _Tp>
_UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ () const
```

Return a new valarray by applying unary  $\sim$  to each element.

The documentation for this class was generated from the following file:

- [valarray](#)

**6.1037 std::\_\_debug::vector< \_Tp, \_Allocator > Class Template Reference**

```
#include <vector>
```

Inheritance diagram for std::\_\_debug::vector< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, vector >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- constexpr `vector` (`_Base_ref __x`)
- template<class `_InputIterator`, typename = `std::_RequireInputIter< _InputIterator >`>>  
constexpr `vector` (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- constexpr `vector` (`const _Allocator &__a`) noexcept
- `vector` (`const vector &`)=default
- constexpr `vector` (`const vector &__x, const __type_identity_t< allocator_type > &__a`)
- constexpr `vector` (`initializer_list< value_type > __l, const allocator_type &__a=allocator_type()`)
- constexpr `vector` (`size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator()`)
- constexpr `vector` (`size_type __n, const _Allocator &__a=_Allocator()`)
- `vector` (`vector &&`)=default
- constexpr `vector` (`vector &&__x, const __type_identity_t< allocator_type > &__a`) noexcept(`std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value`)
- constexpr `const _Base & _M_base` () const noexcept
- constexpr `_Base & _M_base` () noexcept
- template<typename `_InputIterator`, typename = `std::_RequireInputIter< _InputIterator >`>>  
constexpr void `assign` (`_InputIterator __first, _InputIterator __last`)
- constexpr void `assign` (`initializer_list< value_type > __l`)
- constexpr void `assign` (`size_type __n, const _Tp &__u`)
- constexpr `const_reference back` () const noexcept
- constexpr `reference back` () noexcept

- constexpr const\_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size\_type **capacity** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
constexpr iterator **emplace** (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr reference **emplace\_back** (\_Args &&... \_\_args)
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator **erase** (const\_iterator \_\_position)
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>  
constexpr iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Up = \_Tp>  
constexpr \_\_gnu\_cxx::enable\_if<!std::\_\_are\_same<\_Up, bool>::\_\_value, iterator >::\_\_type **insert** (const\_iterator \_\_position, \_Tp &&\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, const \_Tp &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, [initializer\\_list](#)< value\_type > \_\_l)
- constexpr iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- vector & **operator=** (const vector &)=default
- constexpr vector & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- vector & **operator=** (vector &&)=default
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- constexpr reference **operator[]** (size\_type \_\_n) noexcept
- constexpr void **pop\_back** () noexcept
- template<typename \_Up = \_Tp>  
constexpr \_\_gnu\_cxx::enable\_if<!std::\_\_are\_same<\_Up, bool>::\_\_value, void >::\_\_type **push\_back** (\_Tp &&\_\_x)
- constexpr void **push\_back** (const \_Tp &\_\_x)
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr reverse\_iterator **rbegin** () noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept
- constexpr reverse\_iterator **rend** () noexcept
- constexpr void **reserve** (size\_type \_\_n)
- constexpr void **resize** (size\_type \_\_sz)
- constexpr void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- constexpr void **shrink\_to\_fit** ()
- constexpr void **swap** (vector &\_\_x) noexcept([/\\*conditional \\*/](#))

### Protected Member Functions

- bool **\_M\_requires\_reallocation** (size\_type \_\_elements) const noexcept
- constexpr void **\_M\_swap** (const \_Safe\_container &\_\_x) const noexcept
- constexpr void **\_M\_update\_guaranteed\_capacity** () noexcept



**Protected Attributes**

- size\_type **\_M\_guaranteed\_capacity**

**Friends**

- template<typename \_ItT, typename \_SeqT, typename \_CatT>  
class ::**\_\_gnu\_debug::\_Safe\_iterator**

**6.1037.1 Detailed Description**

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::vector< _Tp, _Allocator >
```

Class std::vector with safety/checking/debug instrumentation.

**6.1037.2 Constructor & Destructor Documentation****vector()**

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
std::__debug::vector< _Tp, _Allocator >::vector (
 _Base_ref __x) [inline], [constexpr]
```

Construction from a normal-mode vector.

The documentation for this class was generated from the following file:

- [debug/vector](#)

**6.1038 std::vector< \_Tp, \_Alloc > Class Template Reference**

```
#include <vector>
```

Inheritance diagram for std::vector< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse\_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse\_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [vector](#) ()=default
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
constexpr [vector](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a=allocator\_type())
- constexpr [vector](#) (const `allocator_type` &\_\_a) noexcept
- constexpr [vector](#) (const `vector` &\_\_x)
- constexpr [vector](#) (const `vector` &\_\_x, const `__type_identity_t`< `allocator_type` > &\_\_a)
- constexpr [vector](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `allocator_type` &\_\_a=allocator\_type())
- constexpr [vector](#) (`size_type` \_\_n, const `allocator_type` &\_\_a=allocator\_type())
- constexpr [vector](#) (`size_type` \_\_n, const `value_type` &\_\_value, const `allocator_type` &\_\_a=allocator\_type())
- [vector](#) (`vector` &&) noexcept=default
- constexpr [vector](#) (`vector` &&\_\_rv, const `__type_identity_t`< `allocator_type` > &\_\_m) noexcept(noexcept(`vector`([std::declval](#)< `vector` && >(), [std::declval](#)< const `allocator_type` & >(), [std::declval](#)< typename `_Alloc_traits::is_always_equal` >())))
- constexpr [~vector](#) () noexcept
- template<typename... `_Args`>  
constexpr auto **`_M_emplace_aux`** (const `iterator` \_\_position, `_Args` &&... \_\_args) -> `iterator`
- template<typename `_InputIterator`, typename = `std::_RequireInputIter<_InputIterator>`>>  
constexpr void [assign](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- constexpr void [assign](#) ([initializer\\_list](#)< `value_type` > \_\_l)
- constexpr void [assign](#) (`size_type` \_\_n, const `value_type` &\_\_val)
- constexpr `reference` [at](#) (`size_type` \_\_n)
- constexpr `const_reference` [at](#) (`size_type` \_\_n) const
- constexpr `const_reference` [back](#) () const noexcept
- constexpr `reference` [back](#) () noexcept
- constexpr `const_iterator` [begin](#) () const noexcept
- constexpr `iterator` [begin](#) () noexcept
- constexpr `size_type` [capacity](#) () const noexcept
- constexpr `const_iterator` [cbegin](#) () const noexcept
- constexpr `const_iterator` [cend](#) () const noexcept
- constexpr void [clear](#) () noexcept
- constexpr `const_reverse_iterator` [crbegin](#) () const noexcept
- constexpr `const_reverse_iterator` [crend](#) () const noexcept
- constexpr const `_Tp` \* **`data`** () const noexcept
- constexpr `_Tp` \* [data](#) () noexcept

- template<typename... \_Args>  
constexpr iterator **emplace** (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr reference **emplace\_back** (\_Args &&... \_\_args)
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator **erase** (const\_iterator \_\_position)
- constexpr const\_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr allocator\_type **get\_allocator** () const noexcept
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
constexpr iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, initializer\_list< value\_type > \_\_l)
- constexpr iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- constexpr size\_type **max\_size** () const noexcept
- constexpr vector & **operator=** (const vector &\_\_x)
- constexpr vector & **operator=** (initializer\_list< value\_type > \_\_l)
- constexpr vector & **operator=** (vector &&\_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- constexpr reference **operator[]** (size\_type \_\_n) noexcept
- constexpr void **pop\_back** () noexcept
- constexpr void **push\_back** (const value\_type &\_\_x)
- constexpr void **push\_back** (value\_type &&\_\_x)
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr reverse\_iterator **rbegin** () noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept
- constexpr reverse\_iterator **rend** () noexcept
- constexpr void **reserve** (size\_type \_\_n)
- constexpr void **resize** (size\_type \_\_new\_size)
- constexpr void **resize** (size\_type \_\_new\_size, const value\_type &\_\_x)
- constexpr void **shrink\_to\_fit** ()
- constexpr size\_type **size** () const noexcept
- constexpr void **swap** (vector &\_\_x) noexcept

### Protected Member Functions

- constexpr pointer **\_M\_allocate** (size\_t \_\_n)
- template<typename \_ForwardIterator>  
constexpr \_Alloc\_result **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- constexpr \_Alloc\_result **\_M\_allocate\_at\_least** (size\_t \_\_n)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, std::forward\_iterator\_tag)
- template<typename \_InputIterator>  
constexpr void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, std::input\_iterator\_tag)
- template<typename \_InputIterator>  
constexpr void **\_M\_assign\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
constexpr void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)

- constexpr size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- constexpr void **\_M\_create\_storage** (size\_t \_\_n)
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_default\_append** (size\_type \_\_n)
- constexpr void **\_M\_default\_initialize** (size\_type \_\_n)
- template<typename... \_Args>  
constexpr iterator **\_M\_emplace\_aux** (const\_iterator \_\_position, \_Args &&... \_\_args)
- constexpr iterator **\_M\_emplace\_aux** (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- constexpr iterator **\_M\_erase** (iterator \_\_position)
- constexpr void **\_M\_erase\_at\_end** (pointer \_\_pos) noexcept
- constexpr void **\_M\_fill\_append** (size\_type \_\_n, const value\_type &\_\_x)
- constexpr void **\_M\_fill\_assign** (size\_type \_\_n, const value\_type &\_\_val)
- constexpr void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_value)
- constexpr void **\_M\_fill\_insert** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- constexpr const\_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr\_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- template<typename \_Arg>  
constexpr void **\_M\_insert\_aux** (iterator \_\_position, \_Arg &&\_\_arg)
- template<typename \_InputIterator>  
constexpr void **\_M\_insert\_dispatch** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_Integer>  
constexpr void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- constexpr iterator **\_M\_insert\_rval** (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr void **\_M\_range\_check** (size\_type \_\_n) const
- template<typename \_ForwardIterator>  
constexpr void **\_M\_range\_initialize** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_Iterator, typename \_Sentinel>  
constexpr void **\_M\_range\_initialize\_n** (\_Iterator \_\_first, \_Sentinel \_\_last, size\_type \_\_n)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_range\_insert** (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename... \_Args>  
constexpr void **\_M\_realloc\_append** (\_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr void **\_M\_realloc\_insert** (iterator \_\_position, \_Args &&... \_\_args)
- constexpr void **\_M\_replace\_storage** (pointer \_\_start, pointer \_\_end, size\_type \_\_cap)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_replace\_with** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- constexpr bool **\_M\_shrink\_to\_fit** ()

### Static Protected Member Functions

- static constexpr size\_type **\_S\_check\_init\_len** (size\_type \_\_n, const allocator\_type &\_\_a)
- static constexpr size\_t **\_S\_max\_size** (const\_Tp\_alloc\_type &\_\_a) noexcept

### Protected Attributes

- \_Vector\_impl **\_M\_impl**

### 6.1038.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector<_Tp, _Alloc >
```

A standard container which offers fixed time access to individual elements in any order.

Since

C++98

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

### 6.1038.2 Constructor & Destructor Documentation

#### **vector()** [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector () [default]
```

Creates a vector with no elements.

Referenced by [insert\(\)](#), and [operator=\(\)](#).

#### **vector()** [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector (
 const allocator_type & __a) [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

#### **vector()** [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
| <code>__a</code> | An allocator.                               |

This constructor fills the vector with `__n` default constructed elements.

**vector() [4/10]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

**Parameters**

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the vector with `__n` copies of `__value`.

**vector() [5/10]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 const vector< _Tp, _Alloc > & __x) [inline], [constexpr]
```

Vector copy constructor.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied (i.e. `capacity() == size()` in the new vector).

The newly-created vector uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

**vector() [6/10]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 vector< _Tp, _Alloc > &&) [default], [noexcept]
```

Vector move constructor.

The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

**vector()** [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
 const vector<_Tp, _Alloc> & __x,
 const __type_identity_t< allocator_type > & __a) [inline], [constexpr]
```

Copy constructor with alternative allocator.

**vector()** [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
 vector<_Tp, _Alloc> && __rv,
 const __type_identity_t< allocator_type > & __m) [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

**vector()** [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
 initializer_list< value_type > __l,
 const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from an initializer list.

**Parameters**

|                                       |                      |
|---------------------------------------|----------------------|
| $\leftrightarrow$<br><code>__l</code> | An initializer_list. |
| $\leftrightarrow$<br><code>__a</code> | An allocator.        |

Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor `N` times (where `N` is `__l.size()`) and do no memory reallocation.

**vector()** [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::vector<_Tp, _Alloc>::vector (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline], [constexpr]
```

Builds a vector from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |
| <code>__a</code>     | An allocator.      |

Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

### `~vector()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::~~vector () [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 6.1038.3 Member Function Documentation

### `_M_allocate_and_copy()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator>
_Alloc_result std::vector< _Tp, _Alloc >::_M_allocate_and_copy (
 size_type __n,
 _ForwardIterator __first,
 _ForwardIterator __last) [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain at least  $n$  objects worth of memory, copies `[first, last)` into it, reports what was actually allocated.

Referenced by `reserve()`.

### `_M_range_check()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::_M_range_check (
 size_type __n) const [inline], [constexpr], [protected]
```

Safety check used only from `at()`.

Referenced by `std::vector< _Tp, polymorphic_allocator< _Tp > >::at()`, and `std::vector< _Tp, polymorphic_allocator< _Tp > >::at()`.

### `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector< _Tp, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline], [constexpr]
```

Assigns a range to a vector.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

### `assign()` [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```



```
void std::vector< _Tp, _Alloc >::assign (
 initializer_list< value_type > __l) [inline], [constexpr]
```

Assigns an initializer list to a vector.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| $\leftrightarrow$    | An initializer_list. |
| $\_ \leftrightarrow$ |                      |
| $\leftrightarrow$    |                      |
| $\_ \leftrightarrow$ |                      |
| $/$                  |                      |

This function fills a vector with copies of the elements in the initializer list  $\_l$ .

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

#### assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (
 size_type __n,
 const value_type & __val) [inline], [constexpr]
```

Assigns a given value to a vector.

#### Parameters

|         |                                    |
|---------|------------------------------------|
| $\_n$   | Number of elements to be assigned. |
| $\_val$ | Value to be assigned.              |

This function fills a vector with  $\_n$  copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

#### at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::at (
 size_type __n) [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| $\_ \leftrightarrow$ | The index of the element for which data should be accessed. |
| $\_n$                |                                                             |

**Returns**

Read/write reference to data.

**Exceptions**

|                                                |                                          |
|------------------------------------------------|------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__n</code> is an invalid index. |
|------------------------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

**at()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::at (
 size_type __n) const [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

**Parameters**

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

**Returns**

Read-only (constant) reference to data.

**Exceptions**

|                                                |                                          |
|------------------------------------------------|------------------------------------------|
| <a href="#"><code>std::out_of_range</code></a> | If <code>__n</code> is an invalid index. |
|------------------------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

**back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::back () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

**back()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::back () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

**begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::begin () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

**begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::begin () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Referenced by [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::vector\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::crend\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::empty\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::erase\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::erase\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::front\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::front\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#), [std::operator<=>\(\)](#), [operator=\(\)](#), [std::operator==\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::rend\(\)](#), and [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::rend\(\)](#).

**capacity()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector<_Tp, _Alloc>::capacity () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Referenced by [operator=\(\)](#), and [reserve\(\)](#).

**cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::cbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Referenced by [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::erase\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::erase\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#), [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#), and [std::vector<\\_Tp, polymorphic\\_allocator<\\_Tp>>::insert\(\)](#).

**cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::cend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**clear()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::clear () [inline], [constexpr], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by [operator=\(\)](#).

**crbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::crbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**crend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**data()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp * std::vector< _Tp, _Alloc >::data () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

**emplace()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
iterator std::vector< _Tp, _Alloc >::emplace (
 const_iterator __position,
 _Args &&... __args) [inline], [constexpr]
```

Inserts an object in vector before specified iterator.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the vector. |
| <code>__args</code>     | Arguments.                                     |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

**empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector< _Tp, _Alloc >::empty () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Referenced by `std::piecewise_constant_distribution< _RealType >::densities()`, `std::piecewise_linear_distribution< _RealType >::densities()` and `std::discrete_distribution< _IntType >::probabilities()`.

**end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::end () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

**end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::end () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Referenced by `std::vector< _Tp, polymorphic_allocator< _Tp > >::vector()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::back()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::back()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::crbegin()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::empty()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator<=>()`, `operator=()`, `std::operator==()`, `std::vector< _Tp, polymorphic_allocator< _Tp > >::rbegin()`, and `std::vector< _Tp, polymorphic_allocator< _Tp > >::rbegin()`.

### erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline], [constexpr]
```

Remove a range of elements.

#### Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

#### Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
 const_iterator __position) [inline], [constexpr]
```

Remove element at given position.

#### Parameters

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

#### Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

### front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::front () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

#### **front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

#### **get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::Vector_base< _Tp, _Alloc >::get_allocator () const [inline], [constexpr],
[noexcept]
```

Get a copy of the memory allocation object.

References [get\\_allocator\(\)](#).

Referenced by [get\\_allocator\(\)](#).

#### **insert()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last) [inline], [constexpr]
```

Inserts a range into the vector.

#### **Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__first</code>    | An input iterator.                |
| <code>__last</code>     | An input iterator.                |

#### **Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range [`__first`,`__last`) into the vector before the location specified by *pos*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

#### **insert()** [2/5]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc >::iterator vector::insert (
 const_iterator __position,
 const value_type & __x) [constexpr]
```

Inserts given value into vector before specified iterator.

#### **Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
|-------------------------|-----------------------------------|

|                  |                      |
|------------------|----------------------|
| <code>__x</code> | Data to be inserted. |
|------------------|----------------------|

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

References [vector\(\)](#), [std::begin\(\)](#), [std::cbegin\(\)](#), [std::end\(\)](#), [insert\(\)](#), and [std::move\(\)](#).

Referenced by [insert\(\)](#).

**insert()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::insert (
 const_iterator __position,
 initializer_list< value_type > __l) [inline], [constexpr]
```

Inserts an initializer\_list into the vector.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>__position</code> | An iterator into the vector. |
| <code>__l</code>        | An initializer_list.         |

This function will insert copies of the data in the initializer\_list *l* into the vector before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x) [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the vector.  |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

**Returns**

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**insert()** [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

**max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.

Referenced by [reserve\(\)](#).

**operator=()** [1/3]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc > & vector::operator= (
 const vector< _Tp, _Alloc > & __x) [constexpr]
```

Vector assignment operator.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied.

Whether the allocator is copied depends on the allocator traits.

References [vector\(\)](#), [std::\\_\\_addressof\(\)](#), [std::\\_Destroy\(\)](#), [begin\(\)](#), [capacity\(\)](#), [clear\(\)](#), [std::end\(\)](#), [end\(\)](#), [std::size\(\)](#), and [size\(\)](#).



**operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector<_Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline], [constexpr]
```

Vector list assignment operator.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| $\leftarrow$    | An initializer_list. |
| $\_ \leftarrow$ |                      |
| $\leftarrow$    |                      |
| $\_ \leftarrow$ |                      |
| $/$             |                      |

This function fills a vector with copies of the elements in the initializer list  $\_ /$ .

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

**operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector<_Tp, _Alloc >::operator= (
 vector<_Tp, _Alloc > && __x) [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

**Parameters**

|                 |                                                    |
|-----------------|----------------------------------------------------|
| $\_ \leftarrow$ | A vector of identical element and allocator types. |
| $\_ x$          |                                                    |

The contents of  $\_ x$  are moved into this vector (without copying, if the allocators permit it). Afterwards  $\_ x$  is a valid, but unspecified vector.

Whether the allocator is moved depends on the allocator traits.

**operator[]()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc >::operator[] (
 size_type __n) const [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| $\_ \leftarrow$ | The index of the element for which data should be accessed. |
| $\_ n$          |                                                             |

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out\_of\_range lookups are not defined. (For checked lookups see at().)

**operator[]()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
 size_type __n) [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                  |                                                             |
|--------------------------------------------------|-------------------------------------------------------------|
| <b><code>_↔</code></b><br><b><code>_n</code></b> | The index of the element for which data should be accessed. |
|--------------------------------------------------|-------------------------------------------------------------|

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

**pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back () [inline], [constexpr], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

**push\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::push_back (
 const value_type & __x) [inline], [constexpr]
```

Add data to the end of the vector.

**Parameters**

|                                                  |                   |
|--------------------------------------------------|-------------------|
| <b><code>_↔</code></b><br><b><code>_x</code></b> | Data to be added. |
|--------------------------------------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

**rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc >::rbegin () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

**rend()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc >::rend () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**rend()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc >::rend () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

**reserve()**

```
template<typename _Tp, typename _Alloc>
void vector::reserve (
 size_type __n) [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| $\leftrightarrow$ | Number of elements required. |
| <code>__n</code>  |                              |

**Exceptions**

|                                                |                                                     |
|------------------------------------------------|-----------------------------------------------------|
| <a href="#"><code>std::length_error</code></a> | If <code>n</code> exceeds <code>max_size()</code> . |
|------------------------------------------------|-----------------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

References [std::\\_Destroy\(\)](#), [\\_M\\_allocate\\_and\\_copy\(\)](#), [capacity\(\)](#), [max\\_size\(\)](#), and [std::size\(\)](#).

**resize()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc >::resize (
 size_type __new_size) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

**Parameters**

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_shrink\\_and\\_double\(\)](#), and [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#).

**resize()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
 size_type __new_size,
 const value_type & __x) [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

**shrink\_to\_fit()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::shrink_to_fit () [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

**size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::size () const [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_shrink\(\)](#), [\\_\\_gnu\\_parallel::\\_\\_shrink\\_and\\_double\(\)](#), [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::\\_\\_gnu\\_parallel::list\\_partition\(\)](#), [operator=\(\)](#), [std::operator==\(\)](#), [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::resize\(\)](#), and [std::vector< \\_Tp, polymorphic\\_allocator< \\_Tp > >::resize\(\)](#).

**swap()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::swap (
 vector< _Tp, _Alloc > & __x) [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__↔</code> | A vector of the same element and allocator types. |
| <code>__x</code> |                                                   |

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 6.1039 std::vector< bool, \_Alloc > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::vector< bool, \_Alloc >:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

### Public Member Functions

- template<typename `_InputIterator`, typename `= std::_RequireInputIter<_InputIterator>`>>  
constexpr **vector** (`_InputIterator` `__first`, `_InputIterator` `__last`, `const allocator_type &__a=allocator_type()`)
- constexpr **vector** (`const allocator_type &__a`) noexcept
- constexpr **vector** (`const vector &__x`)
- constexpr **vector** (`const vector &__x`, `const __type_identity_t< allocator_type > &__a`)
- constexpr **vector** (`initializer_list< bool > __l`, `const allocator_type &__a=allocator_type()`)
- constexpr **vector** (`size_type __n`, `const allocator_type &__a=allocator_type()`)
- constexpr **vector** (`size_type __n`, `const bool &__value`, `const allocator_type &__a=allocator_type()`)

- **vector** (vector &&)=default
- constexpr **vector** (vector &&\_\_x, const \_\_type\_identity\_t< allocator\_type > &\_\_a) noexcept(\_Bit\_alloc\_traits::↵\_S\_always\_equal())
- constexpr auto **M\_emplace\_aux** (const\_iterator \_\_position, \_Args &&... \_\_args) -> iterator
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr void **assign** (initializer\_list< bool > \_\_l)
- constexpr void **assign** (size\_type \_\_n, const bool &\_\_x)
- constexpr reference **at** (size\_type \_\_n)
- constexpr const\_reference **at** (size\_type \_\_n) const
- constexpr reference **back** ()
- constexpr const\_reference **back** () const
- constexpr const\_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size\_type **capacity** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const\_reverse\_iterator **crbegin** () const noexcept
- constexpr const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
constexpr iterator **emplace** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
constexpr reference **emplace\_back** (\_Args &&... \_\_args)
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- constexpr iterator **erase** (const\_iterator \_\_position)
- constexpr void **flip** () noexcept
- constexpr reference **front** ()
- constexpr const\_reference **front** () const
- constexpr allocator\_type **get\_allocator** () const
- constexpr allocator\_type **get\_allocator** () const noexcept
- constexpr iterator **insert** (const\_iterator \_\_p, initializer\_list< bool > \_\_l)
- iterator **insert** (const\_iterator \_\_position)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
constexpr iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- constexpr iterator **insert** (const\_iterator \_\_position, const bool &\_\_x)
- constexpr iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- constexpr size\_type **max\_size** () const noexcept
- constexpr vector & **operator=** (const vector &\_\_x)
- constexpr vector & **operator=** (initializer\_list< bool > \_\_l)
- constexpr vector & **operator=** (vector &&\_\_x) noexcept(\_Bit\_alloc\_traits::\_S\_nothrow\_move())
- constexpr reference **operator[]** (size\_type \_\_n)
- constexpr const\_reference **operator[]** (size\_type \_\_n) const
- constexpr void **pop\_back** ()
- constexpr void **push\_back** (bool \_\_x)
- constexpr const\_reverse\_iterator **rbegin** () const noexcept
- constexpr reverse\_iterator **rbegin** () noexcept
- constexpr const\_reverse\_iterator **rend** () const noexcept

- constexpr reverse\_iterator **rend** () noexcept
- constexpr void **reserve** (size\_type \_\_n)
- constexpr void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- constexpr void **shrink\_to\_fit** ()
- constexpr size\_type **size** () const noexcept
- constexpr void **swap** (vector &\_\_x) noexcept

### Static Public Member Functions

- static constexpr size\_t **S\_max\_size** (const Tp\_alloc\_type &\_\_a) noexcept
- static constexpr void **swap** (reference \_\_x, reference \_\_y) noexcept

### Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Bit\_type >::other **\_Bit\_alloc\_type**

### Protected Member Functions

- constexpr pointer **\_M\_allocate** (size\_t \_\_n)
- constexpr \_Alloc\_result **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- constexpr \_Alloc\_result **\_M\_allocate\_at\_least** (size\_t \_\_n)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- constexpr size\_type **\_M\_check\_len** (size\_type \_\_n, const char \*\_\_s) const
- constexpr iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- constexpr void **\_M\_create\_storage** (size\_t \_\_n)
- constexpr void **\_M\_deallocate** ()
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- constexpr void **\_M\_default\_append** (size\_type \_\_n)
- constexpr void **\_M\_default\_initialize** (size\_type \_\_n)
- constexpr iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- constexpr iterator **\_M\_erase** (iterator \_\_pos)
- constexpr void **\_M\_erase\_at\_end** (iterator \_\_pos)
- constexpr void **\_M\_fill\_append** (size\_type \_\_n, const value\_type &\_\_x)
- constexpr void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- constexpr void **\_M\_fill\_initialize** (size\_type \_\_n, const value\_type &\_\_value)
- constexpr void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- constexpr const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const noexcept
- constexpr \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () noexcept
- constexpr const Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- constexpr Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- constexpr void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_ForwardIterator>  
constexpr void **\_M\_initialize\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- template<typename \_InputIterator>  
constexpr void **\_M\_initialize\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_initialize\_value** (bool \_\_x) noexcept
- constexpr void **\_M\_insert\_aux** (iterator \_\_position, bool \_\_x)

- constexpr void **\_M\_insert\_dispatch** (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator>  
constexpr void **\_M\_insert\_range** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator>  
constexpr void **\_M\_insert\_range** (iterator \_\_position, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- constexpr iterator **\_M\_insert\_rval** (const\_iterator \_\_position, value\_type &&\_\_v)
- constexpr void **\_M\_move\_data** (\_Bvector\_base &&\_\_x) noexcept
- constexpr void **\_M\_range\_check** (size\_type \_\_n) const
- constexpr void **\_M\_range\_initialize** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_range\_initialize\_n** (\_Iterator \_\_first, \_Sentinel \_\_last, size\_type \_\_n)
- constexpr void **\_M\_range\_insert** (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- constexpr void **\_M\_realloc\_append** (\_Args &&... \_\_args)
- constexpr void **\_M\_realloc\_insert** (iterator \_\_position, \_Args &&... \_\_args)
- constexpr void **\_M\_reallocate** (size\_type \_\_n)
- constexpr void **\_M\_replace\_storage** (pointer \_\_start, pointer \_\_end, size\_type \_\_cap)
- constexpr void **\_M\_replace\_with** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- constexpr bool **\_M\_shrink\_to\_fit** ()
- void **data** ()=delete

### Static Protected Member Functions

- static constexpr size\_type **\_S\_check\_init\_len** (size\_type \_\_n, const allocator\_type &\_\_a)
- static constexpr size\_t **\_S\_nword** (size\_t \_\_n)

### Protected Attributes

- `_Vector_impl` **\_M\_impl**

### Friends

- struct **std::hash< vector >**

## 6.1039.1 Detailed Description

template<typename **\_Alloc**>  
class **std::vector< bool, \_Alloc >**

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Since

C++98

### Template Parameters

|                     |                 |
|---------------------|-----------------|
| <code>_Alloc</code> | Allocator type. |
|---------------------|-----------------|

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

See also

`vector` for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.



### 6.1039.2 Member Function Documentation

#### **\_M\_allocate\_and\_copy()**

```
_Alloc_result std::vector< bool, _Alloc >::_M_allocate_and_copy (
 size_type __n,
 _ForwardIterator __first,
 _ForwardIterator __last) [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain at least *n* objects worth of memory, copies [*first*, *last*) into it, reports what was actually allocated.

#### **get\_allocator()**

```
() const
```

Get a copy of the memory allocation object.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

## 6.1040 std::ranges::view\_base Struct Reference

```
#include <ranges_base.h>
```

### 6.1040.1 Detailed Description

[range.view] The ranges::view\_base type.

The documentation for this struct was generated from the following file:

- [ranges\\_base.h](#)

## 6.1041 std::ranges::view\_interface< \_Derived > Class Template Reference

```
#include <ranges_util.h>
```

Inheritance diagram for std::ranges::view\_interface< \_Derived >:



### Public Member Functions

- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr auto **cbegin** ()

- constexpr auto **cbegin** () const
- constexpr auto **cend** ()
- constexpr auto **cend** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(\_M\_derived())))
- constexpr bool **empty** () const noexcept(noexcept(\_S\_empty(\_M\_derived())))
- constexpr bool **empty** () const noexcept(noexcept(ranges::size(\_M\_derived())==0))
- constexpr bool **empty** () noexcept(noexcept(\_S\_empty(\_M\_derived())))
- constexpr bool **empty** () noexcept(noexcept(ranges::size(\_M\_derived())==0))
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(\_M\_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(\_M\_derived())))
- template<random\_access\_range \_Range = \_Derived>  
constexpr decltype(auto) **operator[]** (range\_difference\_t< \_Range > \_\_n)
- template<random\_access\_range \_Range = const \_Derived>  
constexpr decltype(auto) **operator[]** (range\_difference\_t< \_Range > \_\_n) const
- constexpr auto **size** () const noexcept(noexcept(\_S\_size(\_M\_derived())))
- constexpr auto **size** () noexcept(noexcept(\_S\_size(\_M\_derived())))

#### 6.1041.1 Detailed Description

```
template<typename _Derived>
requires is_class_v<_Derived> && same_as<_Derived, remove_cv_t<_Derived>>
class std::ranges::view_interface< _Derived >
```

The ranges::view\_interface class template.

The documentation for this class was generated from the following files:

- [ranges\\_base.h](#)
- [ranges\\_util.h](#)

#### 6.1042 std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr > Class Template Reference

```
#include <locale_conv.h>
```

Inheritance diagram for std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >:



## Public Types

- typedef \_Codecvt::state\_type **state\_type**
- typedef \_CharT **char\_type**
- typedef \_Traits **traits\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef basic\_streambuf< char\_type, traits\_type > **\_\_streambuf\_type**

## Public Member Functions

- **wbuffer\_convert** ()
- **wbuffer\_convert** (const wbuffer\_convert &)=delete
- **wbuffer\_convert** (streambuf \* \_\_bytebuf, \_Codecvt \* \_\_pcvt=new \_Codecvt, state\_type \_\_state=state\_type())
- **locale getloc** () const
- **streamsize in\_avail** ()
- **wbuffer\_convert & operator=** (const wbuffer\_convert &)=delete
- **locale pubimbue** (const locale & \_\_loc)
- **streambuf \* rdbuf** () const noexcept
- **streambuf \* rdbuf** (streambuf \* \_\_bytebuf) noexcept
- **int\_type sbumpc** ()
- **int\_type sgetc** ()
- **streamsize sgetn** (char\_type \* \_\_s, streamsize \_\_n)
- **int\_type snextc** ()
- **int\_type sputbackc** (char\_type \_\_c)
- **int\_type sputc** (char\_type \_\_c)
- **streamsize sputn** (const char\_type \* \_\_s, streamsize \_\_n)
- **state\_type state** () const noexcept
- **int\_type sungetc** ()
- **basic\_streambuf \* pubsetbuf** (char\_type \* \_\_s, streamsize \_\_n)
- **pos\_type pubseekoff** (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **pos\_type pubseekpos** (pos\_type \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
- **int pubsync** ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
- void **\_\_safe\_pbump** (streamsize \_\_n)
- void **gbump** (int \_\_n)
- virtual void **imbue** (const locale & \_\_loc)
- **\_Wide\_streambuf::int\_type overflow** (typename \_Wide\_streambuf::int\_type \_\_out)
- virtual **int\_type pbackfail** (int\_type \_\_c=traits\_type::eof())
- void **pbump** (int \_\_n)
- virtual **pos\_type seekoff** (off\_type, ios\_base::seekdir, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual **pos\_type seekpos** (pos\_type, ios\_base::openmode=ios\_base::in|ios\_base::out)
- virtual **basic\_streambuf< char\_type, \_Traits > \* setbuf** (char\_type \*, streamsize)

- void `setg` (`char_type` \* \_\_gbeg, `char_type` \* \_\_gnext, `char_type` \* \_\_gend)
  - void `setp` (`char_type` \* \_\_pbeg, `char_type` \* \_\_pend)
  - virtual `streamsize showmanyc` ()
  - void `swap` (`basic_streambuf` & \_\_sb)
  - int `sync` ()
  - virtual `int_type uflow` ()
  - `_Wide_streambuf::int_type underflow` ()
  - virtual `streamsize xsgetn` (`char_type` \* \_\_s, `streamsize` \_\_n)
  - virtual `streamsize xspn` (const `char_type` \* \_\_s, `streamsize` \_\_n)
  - `streamsize xspn` (const typename `_Wide_streambuf::char_type` \* \_\_s, `streamsize` \_\_n)
- 
- `char_type` \* `eback` () const
  - `char_type` \* `gptr` () const
  - `char_type` \* `egptr` () const
- 
- `char_type` \* `pbase` () const
  - `char_type` \* `pptr` () const
  - `char_type` \* `epptr` () const

### Protected Attributes

- `locale _M_buf_locale`
- `char_type` \* `_M_in_beg`
- `char_type` \* `_M_in_cur`
- `char_type` \* `_M_in_end`
- `char_type` \* `_M_out_beg`
- `char_type` \* `_M_out_cur`
- `char_type` \* `_M_out_end`

### 6.1042.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
```

Buffer conversions.

### 6.1042.2 Member Typedef Documentation

#### `__streambuf_type`

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__streambuf_type [inherited]
This is a non-standard type.
```

#### `char_type`

```
template<typename _CharT, typename _Traits>
typedef std::basic_streambuf< _CharT, _Traits >::char_type [inherited]
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.
```

**int\_type**

```
template<typename _CharT, typename _Traits>
```

```
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**off\_type**

```
template<typename _CharT, typename _Traits>
```

```
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**pos\_type**

```
template<typename _CharT, typename _Traits>
```

```
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**traits\_type**

```
template<typename _CharT, typename _Traits>
```

```
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

**6.1042.3 Constructor & Destructor Documentation****wbuffer\_convert() [1/2]**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
```

```
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert () [inline]
```

Default constructor.

References [wbuffer\\_convert\(\)](#).

Referenced by [wbuffer\\_convert\(\)](#).

**wbuffer\_convert() [2/2]**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
```

```
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
```

```
 __streambuf * __bytebuf,
```

```
 _Codecvt * __pcvt = new _Codecvt,
```

```
 state_type __state = state_type()) [inline], [explicit]
```

Constructor.

**Parameters**

|                        |                                    |
|------------------------|------------------------------------|
| <code>__bytebuf</code> | The underlying byte stream buffer. |
| <code>__pcvt</code>    | The facet to use for conversions.  |
| <code>__state</code>   | Initial conversion state.          |

Takes ownership of `__pcvt` and will delete it in the destructor.

References [std::basic\\_streambuf< \\_CharT, \\_Traits >::setg\(\)](#), and [std::basic\\_streambuf< \\_CharT, \\_Traits >::setp\(\)](#).

## 6.1042.4 Member Function Documentation

### eback()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

### egptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::M_create_pback()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char_type, traits_type >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

### epptr()

```
template<typename _CharT, typename _Traits>
```

```
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- ep\_ptr() returns the end pointer for the output sequence

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, `std::basic_streambuf< char_type, traits_type >::sputc()`, `std::basic_filebuf< _CharT, _Traits >::xsputn()`, and `xsputn()`.

**gbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Referenced by `std::basic_filebuf< _CharT, _Traits >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_streambuf< char_type, traits_type >::sbumpc()`, `std::basic_streambuf< char_type, traits_type >::sputbackc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, and `std::basic_filebuf< _CharT, _Traits >::xsgetn()`.

**getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**gptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Referenced by `std::basic_filebuf< char_type, traits_type >::_M_create_pback()`, `std::basic_filebuf< char_type, traits_type >::_M_destroy`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_streambuf< char_type, traits_type >::in_avail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_streambuf< char_type, traits_type >::sgetc()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< char >::showmanyc()`, `std::basic_streambuf< char_type, traits_type >::sungetc()`, `std::basic_streambuf< char_type, traits_type >::uflow()`, `std::basic_filebuf< _CharT, _Traits >::underflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow()`, `std::basic_filebuf< _CharT, _Traits >::xsgetn()`, and `xsgetn()`.

**imbue()**

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< _CharT, std::char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_filebuf< wchar_t >`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubimbue()`.

**in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

**overflow()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow (
 typename _Wide_streambuf::int_type __c) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).



**Note**

Base class version does nothing, returns eof().

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

References [std::basic\\_streambuf<\\_CharT, \\_Traits >::sputc\(\)](#).

**pbackfail()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits >::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <a href="#">_C</a> | The character to be inserted back into the sequence. |
|--------------------|------------------------------------------------------|

**Returns**

eof() on failure, *some other value* on success

**Postcondition**

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

**Note**

Base class version does nothing, returns eof().

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits >](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >](#), [std::basic\\_filebuf<char >](#), [std::basic\\_filebuf<char\\_type, traits\\_type >](#), [std::basic\\_filebuf<wchar\\_t >](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits >](#), [std::basic\\_stringbuf<char >](#), and [std::basic\\_stringbuf<wchar\\_t >](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type >::sputbackc\(\)](#), and [std::basic\\_streambuf<char\\_type, traits\\_type >::sungetc\(\)](#).

**pbase()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf<_CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits >::str\(\)](#), [std::basic\\_filebuf<\\_CharT, \\_Traits >::sync\(\)](#), and [std::basic\\_filebuf<\\_CharT, \\_Traits >::xsputn\(\)](#).

**pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), and [std::basic\\_streambuf< char\\_type, traits\\_type >::sputc\(\)](#).

**pptr()**

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Referenced by [std::basic\\_filebuf< \\_CharT, \\_Traits >::overflow\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::overflow\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::seekoff\(\)](#), [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >::seekoff\(\)](#), [std::basic\\_streambuf< char\\_type, traits\\_type >::seekoff\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::sync\(\)](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >::xsputn\(\)](#), and [xsputn\(\)](#).

**pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived [imbue\(\\_\\_loc\)](#).

**pubseekoff()**

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
```

```

 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]

```

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

#### pubseekpos()

```

template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]

```

Alters the stream position.

#### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

#### pubsetbuf()

```

template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]

```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

#### pubsync()

```

template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]

```

Calls virtual sync function.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

#### sbumpc()

```

template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]

```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**seekoff()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits >`, `std::basic_stringbuf< char >`, and `std::basic_stringbuf< wchar_t >`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubseekoff()`.

**seekpos()**

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_filebuf< wchar_t >`, `std::basic_stringbuf< _CharT, _Traits >`, `std::basic_stringbuf< char >`, and `std::basic_stringbuf< wchar_t >`.

Referenced by `std::basic_streambuf< char_type, traits_type >::pubseekpos()`.

**setbuf()**

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
 char_type * ,
 streamsize) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

**Note**

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<_CharT, std::char_traits<_CharT>>`, `std::basic_filebuf<char>`, `std::basic_filebuf<char_type, traits_type>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, and `std::basic_stringbuf<char>`.

Referenced by `std::basic_streambuf<char_type, traits_type>::pubsetbuf()`.

**setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, `std::basic_filebuf<char_type, traits_type>::M_create()`, `std::basic_filebuf<char_type, traits_type>::M_destroy_pback()`, `std::basic_filebuf<char_type, traits_type>::M_set_buffer()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_filebuf<_CharT, _Traits>::xsgetn()`.

**setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::wbuffer_convert()`, and `std::basic_filebuf<char_type, traits_type>::M_s`

**sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Referenced by `std::basic_istream< char_type, traits_type >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::ignore()` and `std::basic_streambuf< char_type, traits_type >::snextc()`.

**sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

**showmanyc()**

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc () [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_filebuf<wchar\\_t>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), [std::basic\\_stringbuf<char>](#), and [std::basic\\_stringbuf<wchar\\_t>](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::in\\_avail\(\)](#).

### snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Referenced by [std::basic\\_iostream<char\\_type, traits\\_type>::get\(\)](#), [std::basic\\_istream<\\_CharT, \\_Traits>::get\(\)](#), [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::get\(\)](#), and [std::basic\\_istream<\\_CharT, char\\_traits<\\_CharT>>::readsome\(\)](#).

### sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <a href="#">_↔</a> | The character to push back. |
| <a href="#">_c</a> |                             |

#### Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Referenced by [std::basic\\_istream<\\_CharT, \\_Traits>::putback\(\)](#).

### sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

#### Parameters

|                    |                        |
|--------------------|------------------------|
| <a href="#">_↔</a> | A character to output. |
| <a href="#">_c</a> |                        |

**Returns**

\_\_c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores \_\_c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵ __c)`.

Referenced by `std::wbuffer_convert<_Codecv, _Elem, _Tr>::overflow()`.

**sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|          |                     |
|----------|---------------------|
| ↵<br>__s | A buffer read area. |
| ↵<br>__n | A count.            |

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

**state()**

```
template<typename _Codecv, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert<_Codecv, _Elem, _Tr>::state () const [inline], [noexcept]
```

The conversion state following the last conversion.

**sungetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

**sync()**

```
template<typename _Codecv, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert<_Codecv, _Elem, _Tr>::sync () [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.



**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

**uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits >](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type >::sbumpc\(\)](#), and [xsgetn\(\)](#).

**underflow()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert<_Codecvt, _Elem, _Tr >::underflow () [inline],
[protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits >](#).

References [std::basic\\_streambuf<\\_CharT, \\_Traits >::egptr\(\)](#), and [std::basic\\_streambuf<\\_CharT, \\_Traits >::gptr\(\)](#).

**xsgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

#### Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

References [egptr\(\)](#), [gptra\(\)](#), [std::min\(\)](#), and [uflow\(\)](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::sgetn\(\)](#).

#### `xspn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xspn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

#### Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

#### Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<\\_CharT, std::char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

References [eptra\(\)](#), [std::min\(\)](#), [overflow\(\)](#), and [pptra\(\)](#).

Referenced by [std::basic\\_streambuf<char\\_type, traits\\_type>::spn\(\)](#).

### 6.1042.5 Member Data Documentation

#### `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Referenced by [std::basic\\_filebuf<\\_CharT, \\_Traits>::basic\\_filebuf\(\)](#).

### **\_M\_in\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg [protected], [inherited]
Start of get area.
```

### **\_M\_in\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur [protected], [inherited]
Current read area.
```

### **\_M\_in\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end [protected], [inherited]
End of get area.
```

### **\_M\_out\_beg**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
Start of put area.
```

### **\_M\_out\_cur**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
Current put area.
```

### **\_M\_out\_end**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_end [protected], [inherited]
End of put area.
```

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## **6.1043 std::weak\_ptr<\_Tp> Class Template Reference**

```
#include <memory>
```

### **Public Types**

- using **element\_type**

### **Public Member Functions**

- template<typename \_Yp, typename = \_Constructible<const shared\_ptr<\_Yp>&>>  
  **weak\_ptr** (const [shared\\_ptr](#)<\_Yp> &\_\_r) noexcept
- **weak\_ptr** (const weak\_ptr &) noexcept=default

- `template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>>`  
`weak_ptr (const weak_ptr<_Yp> &__r) noexcept`
- `weak_ptr (weak_ptr &&) noexcept=default`
- `template<typename _Yp, typename = _Constructible<weak_ptr<_Yp>>>>`  
`weak_ptr (weak_ptr<_Yp> &&__r) noexcept`
- `bool expired () const noexcept`
- `shared_ptr<_Tp> lock () const noexcept`
- `template<typename _Yp>`  
`_Assignable< const shared_ptr<_Yp> &> operator= (const shared_ptr<_Yp> &__r) noexcept`
- `weak_ptr & operator= (const weak_ptr &__r) noexcept=default`
- `template<typename _Yp>`  
`_Assignable< const weak_ptr<_Yp> &> operator= (const weak_ptr<_Yp> &__r) noexcept`
- `weak_ptr & operator= (weak_ptr &&__r) noexcept=default`
- `template<typename _Yp>`  
`_Assignable< weak_ptr<_Yp> > operator= (weak_ptr<_Yp> &&__r) noexcept`
- `template<typename _Tp1>`  
`bool owner_before (const __shared_ptr<_Tp1, _Lp> &__rhs) const noexcept`
- `template<typename _Tp1>`  
`bool owner_before (const __weak_ptr<_Tp1, _Lp> &__rhs) const noexcept`
- `void reset () noexcept`
- `void swap (__weak_ptr &__s) noexcept`
- `long use_count () const noexcept`

## Related Symbols

(Note that these are not member symbols.)

- `template<typename _Tp>`  
`void swap (weak_ptr<_Tp> &__a, weak_ptr<_Tp> &__b) noexcept`

### 6.1043.1 Detailed Description

```
template<typename _Tp>
class std::weak_ptr<_Tp>
```

A non-owning observer for a pointer owned by a `shared_ptr`.

Since

C++11

A `weak_ptr` provides a safe alternative to a raw pointer when you want a non-owning reference to an object that is managed by a `shared_ptr`.

Unlike a raw pointer, a `weak_ptr` can be converted to a new `shared_ptr` that shares ownership with every other `shared_ptr` that already owns the pointer. In other words you can upgrade from a non-owning "weak" reference to an owning `shared_ptr`, without having access to any of the existing `shared_ptr` objects.

Also unlike a raw pointer, a `weak_ptr` does not become "dangling" after the object it points to has been destroyed. Instead, a `weak_ptr` becomes *expired* and can no longer be converted to a `shared_ptr` that owns the freed pointer, so you cannot accidentally access the pointed-to object after it has been destroyed.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 6.1044 std::weibull\_distribution<\_RealType> Class Template Reference

```
#include <random>
```

**Classes**

- struct [param\\_type](#)

**Public Types**

- typedef \_RealType [result\\_type](#)

**Public Member Functions**

- **weibull\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator>  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator>  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator>  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

**Friends**

- bool **operator==** (const weibull\_distribution &\_\_d1, const weibull\_distribution &\_\_d2)

**6.1044.1 Detailed Description**

**template<typename \_RealType = double>**  
**class std::weibull\_distribution<\_RealType>**

A weibull\_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)$$

Since

C++11

### 6.1044.2 Member Typedef Documentation

#### result\_type

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

### 6.1044.3 Member Function Documentation

#### a()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a () const [inline]
```

Return the  $a$  parameter of the distribution.

#### b()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::b () const [inline]
```

Return the  $b$  parameter of the distribution.

#### max()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

#### min()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

#### operator>()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::weibull_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Referenced by [operator>\(\)\(\)](#).

#### param() [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Referenced by [std::operator>>\(\)](#).

#### param() [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

**reset()**

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset () [inline]
Resets the distribution state.
```

**6.1044.4 Friends And Related Symbol Documentation****operator==**

```
template<typename _RealType = double>
bool operator== (
 const weibull_distribution< _RealType > & __d1,
 const weibull_distribution< _RealType > & __d2) [friend]
```

Return true if two Weibull distributions have the same parameters.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**6.1045 std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc > Class Template Reference**

```
#include <locale_conv.h>
```

**Public Types**

- typedef [basic\\_string](#)< char, [char\\_traits](#)< char >, \_Byte\_alloc > **byte\_string**
- typedef [wide\\_string::traits\\_type::int\\_type](#) **int\_type**
- typedef [\\_Codecvt::state\\_type](#) **state\_type**
- typedef [basic\\_string](#)< \_Elem, [char\\_traits](#)< \_Elem >, \_Wide\_alloc > **wide\_string**

**Public Member Functions**

- [wstring\\_convert](#) ()
- [wstring\\_convert](#) (\_Codecvt \* \_\_pcvt)
- [wstring\\_convert](#) (\_Codecvt \* \_\_pcvt, state\_type \_\_state)
- [wstring\\_convert](#) (const byte\_string & \_\_byte\_err, const wide\_string & \_\_wide\_err=wide\_string())
- **wstring\_convert** (const wstring\_convert &)=delete
- size\_t [converted](#) () const noexcept
- [wstring\\_convert](#) & **operator=** (const [wstring\\_convert](#) &)=delete
- state\_type [state](#) () const
- wide\_string [from\\_bytes](#) (char \_\_byte)
- wide\_string [from\\_bytes](#) (const char \* \_\_ptr)
- wide\_string [from\\_bytes](#) (const byte\_string & \_\_str)
- wide\_string [from\\_bytes](#) (const char \* \_\_first, const char \* \_\_last)

- `byte_string` [to\\_bytes](#) (`_Elem __wchar`)
- `byte_string` [to\\_bytes](#) (`const _Elem * __ptr`)
- `byte_string` [to\\_bytes](#) (`const wide_string & __wstr`)
- `byte_string` [to\\_bytes](#) (`const _Elem * __first, const _Elem * __last`)

### 6.1045.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, type-
name _Byte_alloc = allocator<char>>>
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >
```

String conversions.

### 6.1045.2 Constructor & Destructor Documentation

#### `wstring_convert()` [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert () [inline]
Default constructor.
```

#### `wstring_convert()` [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 _Codecvt * __pcvt) [inline], [explicit]
```

Constructor.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__pcvt</code> | The facet to use for conversions. |
|---------------------|-----------------------------------|

Takes ownership of `__pcvt` and will delete it in the destructor.

#### `wstring_convert()` [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 _Codecvt * __pcvt,
 state_type __state) [inline]
```

Construct with an initial conversion state.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__pcvt</code>  | The facet to use for conversions. |
| <code>__state</code> | Initial conversion state.         |

Takes ownership of `__pcvt` and will delete it in the destructor. The object's conversion state will persist between conversions.



**wstring\_convert()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 const byte_string & __byte_err,
 const wide_string & __wide_err = wide_string()) [inline], [explicit]
```

Construct with error strings.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__byte_err</code> | A string to return on failed conversions.      |
| <code>__wide_err</code> | A wide string to return on failed conversions. |

**6.1045.3 Member Function Documentation****converted()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted () const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

**from\_bytes()** [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 char __byte) [inline]
```

Convert from bytes.

References [from\\_bytes\(\)](#).

Referenced by [from\\_bytes\(\)](#), [from\\_bytes\(\)](#), and [from\\_bytes\(\)](#).

**from\_bytes()** [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const byte_string & __str) [inline]
```

Convert from bytes.

References [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::data\(\)](#), [from\\_bytes\(\)](#), and [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>::size\(\)](#).

**from\_bytes()** [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const char * __first,
 const char * __last) [inline]
```

Convert from bytes.

**from\_bytes()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const char * __ptr) [inline]
```

Convert from bytes.

References [from\\_bytes\(\)](#).

**state()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state () const
[inline]
```

The final conversion state of the last conversion.

**to\_bytes()** [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 _Elem __wchar) [inline]
```

Convert to bytes.

References [to\\_bytes\(\)](#).

Referenced by [to\\_bytes\(\)](#), [to\\_bytes\(\)](#), and [to\\_bytes\(\)](#).

**to\_bytes()** [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const _Elem * __first,
 const _Elem * __last) [inline]
```

Convert to bytes.

**to\_bytes()** [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const _Elem * __ptr) [inline]
```

Convert to bytes.

References [std::basic\\_string< \\_Elem, char\\_traits< \\_Elem >, \\_Wide\\_alloc >::length\(\)](#), and [to\\_bytes\(\)](#).

**to\_bytes()** [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const wide_string & __wstr) [inline]
```

Convert to bytes.

References [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::data\(\)](#), [std::basic\\_string< \\_CharT, \\_Traits, \\_Alloc >::size\(\)](#), and [to\\_bytes\(\)](#).

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## 7 File Documentation

### 7.1 algorithm File Reference

#### 7.1.1 Detailed Description

This is a Standard C++ Library header.

### 7.2 experimental/algorithm File Reference

#### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

#### Macros

- #define `__cpp_lib_experimental_sample`
- #define `_GLIBCXX_EXPERIMENTAL_ALGORITHM`

#### Functions

- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>`  
`_SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>`  
`_SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator std::experimental::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator>`  
`void std::experimental::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`

#### 7.2.1 Detailed Description

This is a TS C++ Library header.

### 7.3 ext/algorithm File Reference

#### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### Macros

- #define `_EXT_ALGORITHM`

#### Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > \_\_gnu\_cxx::\_\_copy\_n (_InputIterator __first, _Size __count, _OutputIterator __result, std::input\_iterator\_tag)`

- `template<typename _RAIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _RAIterator, _OutputIterator > gnu_cxx::copy_n (_RAIterator __first, _Size __count, _OutputIterator __result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `int gnu_cxx::lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `int gnu_cxx::lexicographical_compare_3way (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)`
- `template<typename _Tp>`  
`const _Tp & gnu_cxx::median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare>`  
`const _Tp & gnu_cxx::median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`std::pair< _InputIterator, _OutputIterator > gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size>`  
`void gnu_cxx::count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size>`  
`void gnu_cxx::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _RAIter>`  
`constexpr bool gnu_cxx::is_heap (_RAIter, _RAIter)`
- `template<typename _Filter>`  
`constexpr bool gnu_cxx::is_sorted (_Filter, _Filter)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`int gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`_RandomAccessIterator gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>`  
`_OutputIterator gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>`  
`_OutputIterator gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`

### 7.3.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.4 parallel/algorithm File Reference

### Macros

- `#define _PARALLEL_ALGORITHM`

#### 7.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.5 any File Reference

### 7.5.1 Detailed Description

This is a Standard C++ Library header.

## 7.6 experimental/any File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::any](#)
- class [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_any`
- `#define _GLIBCXX_EXPERIMENTAL_ANY`

### Functions

- `template<typename _ValueType>`  
`_ValueType std::experimental::any_cast (const any &__any)`
- `void std::experimental::swap (any &__x, any &__y) noexcept`
- `template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>`  
`_ValueType std::experimental::any_cast (any &&__any)`
- `template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>`  
`_ValueType std::experimental::any_cast (any &&__any)`
- `template<typename _ValueType>`  
`_ValueType std::experimental::any_cast (any &__any)`
- `template<typename _ValueType>`  
`_ValueType * std::experimental::any_cast (any *__any) noexcept`
- `template<typename _ValueType>`  
`const _ValueType * std::experimental::any_cast (const any *__any) noexcept`

### 7.6.1 Detailed Description

This is a TS C++ Library header.

## 7.7 array File Reference

### Classes

- struct [std::array< \\_Tp, \\_Nm >](#)
- struct [std::tuple\\_element< \\_Ind, array< \\_Tp, \\_Nm > >](#)
- struct [std::tuple\\_size< array< \\_Tp, \\_Nm > >](#)

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename... _Up>  
std::array(_Tp, _Up...) -> array< enable\_if\_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...( _Up)>`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && std::get(array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & std::get(array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && std::get(const array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & std::get(const array< _Tp, _Nm > &__arr) noexcept`
- `template<typename _Tp, size_t _Nm>  
constexpr __detail::__synth3way_t< _Tp > std::operator<=>(const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
__enable_if_t<!__array_traits< _Tp, _Nm >::is_swappable::value > std::swap(array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, std::size_t _Nm>  
constexpr __enable_if_t< __array_traits< _Tp, _Nm >::is_swappable::value > std::swap(array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`

### Variables

- `template<typename _Tp, size_t _Nm>  
constexpr size_t std::tuple_size_v< array< _Tp, _Nm > >`
- `template<typename _Tp, size_t _Nm>  
constexpr size_t std::tuple_size_v< const array< _Tp, _Nm > >`

### 7.7.1 Detailed Description

This is a Standard C++ Library header.

## 7.8 experimental/array File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- #define [\\_\\_cpp\\_lib\\_experimental\\_make\\_array](#)
- #define [\\_GLIBCXX\\_EXPERIMENTAL\\_ARRAY](#)

### Functions

- template<typename [\\_Tp](#), size\_t [\\_Nm](#), size\_t... [\\_Idx](#)>  
constexpr [array](#)< [remove\\_cv\\_t](#)< [\\_Tp](#) >, [\\_Nm](#) > [std::experimental::\\_\\_to\\_array](#) ([\\_Tp](#)(&[\\_\\_a](#))[[\\_Nm](#)],  
, [index\\_sequence](#)< [\\_Idx](#)... >)
- template<typename [\\_Dest](#) = void, typename... [\\_Types](#)>  
constexpr [array](#)< typename [\\_\\_make\\_array\\_elem](#)< [\\_Dest](#), [\\_Types](#)... >::type, sizeof...([\\_Types](#))> [std::experimental::make\\_array](#)  
([\\_Types](#) &&... [\\_\\_t](#))
- template<typename [\\_Tp](#), size\_t [\\_Nm](#)>  
constexpr [array](#)< [remove\\_cv\\_t](#)< [\\_Tp](#) >, [\\_Nm](#) > [std::experimental::to\\_array](#) ([\\_Tp](#)(&[\\_\\_a](#))[[\\_Nm](#)]) noexcept([is\\_nothrow\\_constructible](#)<  
[remove\\_cv\\_t](#)< [\\_Tp](#) >, [\\_Tp](#) & >::value)

#### 7.8.1 Detailed Description

This is a TS C++ Library header.

## 7.9 atomic File Reference

### Classes

- struct [std::atomic](#)< [\\_Tp](#) >
- struct [std::atomic](#)< [\\_Tp](#) \* >
- struct [std::atomic](#)< [bool](#) >
- struct [std::atomic](#)< [char](#) >
- struct [std::atomic](#)< [char16\\_t](#) >
- struct [std::atomic](#)< [char32\\_t](#) >
- struct [std::atomic](#)< [int](#) >
- struct [std::atomic](#)< [long](#) >
- struct [std::atomic](#)< [long long](#) >
- struct [std::atomic](#)< [short](#) >
- struct [std::atomic](#)< [signed char](#) >
- struct [std::atomic](#)< [unsigned char](#) >
- struct [std::atomic](#)< [unsigned int](#) >
- struct [std::atomic](#)< [unsigned long](#) >
- struct [std::atomic](#)< [unsigned long long](#) >
- struct [std::atomic](#)< [unsigned short](#) >
- struct [std::atomic](#)< [wchar\\_t](#) >

### Namespaces

- namespace [std](#)

## Typedefs

- typedef [atomic](#)< bool > [std::atomic\\_bool](#)
- typedef [atomic](#)< char > [std::atomic\\_char](#)
- typedef [atomic](#)< char16\_t > [std::atomic\\_char16\\_t](#)
- typedef [atomic](#)< char32\_t > [std::atomic\\_char32\\_t](#)
- typedef [atomic](#)< int > [std::atomic\\_int](#)
- typedef [atomic](#)< int16\_t > [std::atomic\\_int16\\_t](#)
- typedef [atomic](#)< int32\_t > [std::atomic\\_int32\\_t](#)
- typedef [atomic](#)< int64\_t > [std::atomic\\_int64\\_t](#)
- typedef [atomic](#)< int8\_t > [std::atomic\\_int8\\_t](#)
- typedef [atomic](#)< int\_fast16\_t > [std::atomic\\_int\\_fast16\\_t](#)
- typedef [atomic](#)< int\_fast32\_t > [std::atomic\\_int\\_fast32\\_t](#)
- typedef [atomic](#)< int\_fast64\_t > [std::atomic\\_int\\_fast64\\_t](#)
- typedef [atomic](#)< int\_fast8\_t > [std::atomic\\_int\\_fast8\\_t](#)
- typedef [atomic](#)< int\_least16\_t > [std::atomic\\_int\\_least16\\_t](#)
- typedef [atomic](#)< int\_least32\_t > [std::atomic\\_int\\_least32\\_t](#)
- typedef [atomic](#)< int\_least64\_t > [std::atomic\\_int\\_least64\\_t](#)
- typedef [atomic](#)< int\_least8\_t > [std::atomic\\_int\\_least8\\_t](#)
- typedef [atomic](#)< intmax\_t > [std::atomic\\_intmax\\_t](#)
- typedef [atomic](#)< intptr\_t > [std::atomic\\_intptr\\_t](#)
- typedef [atomic](#)< long long > [std::atomic\\_llong](#)
- typedef [atomic](#)< long > [std::atomic\\_long](#)
- typedef [atomic](#)< ptrdiff\_t > [std::atomic\\_ptrdiff\\_t](#)
- typedef [atomic](#)< signed char > [std::atomic\\_schar](#)
- typedef [atomic](#)< short > [std::atomic\\_short](#)
- typedef [atomic](#)< size\_t > [std::atomic\\_size\\_t](#)
- typedef [atomic](#)< unsigned char > [std::atomic\\_uchar](#)
- typedef [atomic](#)< unsigned int > [std::atomic\\_uint](#)
- typedef [atomic](#)< uint16\_t > [std::atomic\\_uint16\\_t](#)
- typedef [atomic](#)< uint32\_t > [std::atomic\\_uint32\\_t](#)
- typedef [atomic](#)< uint64\_t > [std::atomic\\_uint64\\_t](#)
- typedef [atomic](#)< uint8\_t > [std::atomic\\_uint8\\_t](#)
- typedef [atomic](#)< uint\_fast16\_t > [std::atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic](#)< uint\_fast32\_t > [std::atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic](#)< uint\_fast64\_t > [std::atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic](#)< uint\_fast8\_t > [std::atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [std::atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [std::atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [std::atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [std::atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [std::atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [std::atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [std::atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [std::atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [std::atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [std::atomic\\_wchar\\_t](#)



## Functions

- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_add_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`

- `template<typename _ITp>`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`

- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const atomic<_ITp> *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`_ITp std::atomic_load_explicit (const volatile atomic<_ITp> *__a, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp>`  
`void std::atomic_store_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`

### 7.9.1 Detailed Description

This is a Standard C++ Library header.

## 7.10 auto\_ptr.h File Reference

### Classes

- class `std::auto_ptr<_Tp>`
- struct `std::auto_ptr_ref<_Tp1>`

### Namespaces

- namespace `std`

### 7.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.11 backward\_warning.h File Reference

### 7.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.12 hash\_fun.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Typedefs

- typedef `__SIZE_TYPE__ __gnu_cxx::size_t`

### Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

### 7.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.13 hash\_map File Reference

### Classes

- class [\\_\\_gnu\\_cxx::allocator<\\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::equal\\_to<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- struct [\\_\\_gnu\\_cxx::pair<\\_T1, \\_T2>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Macros

- `#define \_BACKWARD\_HASH\_MAP`

### Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>  
bool \_\_gnu\_cxx::operator!= (const hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm1, const hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>  
bool \_\_gnu\_cxx::operator!= (const hash\_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm1, const hash\_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>  
bool \_\_gnu\_cxx::operator== (const hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm1, const hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>  
bool \_\_gnu\_cxx::operator== (const hash\_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm1, const hash\_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>  
void \_\_gnu\_cxx::swap (hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm1, hash\_map<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc>  
void \_\_gnu\_cxx::swap (hash\_multimap<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm1, hash\_multimap<_Key, _Tp, _HashFn, _EqKey, _Alloc> &__hm2)`

### 7.13.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.14 hash\_set File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_set<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#)

**Namespaces**

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

**Macros**

- `#define` **\_BACKWARD\_HASH\_SET**

**Functions**

- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool` **\_\_gnu\_cxx::operator!=** (const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool` **\_\_gnu\_cxx::operator!=** (const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool` **\_\_gnu\_cxx::operator==** (const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool` **\_\_gnu\_cxx::operator==** (const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, const [hash\\_set](#)< \_Value, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void` **\_\_gnu\_cxx::swap** ([hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, [hash\\_multiset](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`void` **\_\_gnu\_cxx::swap** ([hash\\_set](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs1, [hash\\_set](#)< \_Val, \_HashFcn, \_EqualKey, \_Alloc > &\_\_hs2)

**7.14.1 Detailed Description**

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

**7.15 strstream File Reference****Namespaces**

- namespace [std](#)

**Macros**

- `#define` **\_BACKWARD\_STRSTREAM**
- `#define` **\_GLIBCXX\_STRSTREAM\_DEPR**(A, B)

**7.15.1 Detailed Description**

This is a Standard C++ Library header.

**7.16 barrier File Reference****7.16.1 Detailed Description**

This is a Standard C++ Library header.

## 7.17 bit File Reference

### Namespaces

- namespace [std](#)

#### 7.17.1 Detailed Description

This is a Standard C++ Library header.

## 7.18 bits/algorithmfwd.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Filter>`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2>`  
`constexpr _BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter>`  
`constexpr _OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp = typename iterator_traits<_Iter>::value_type>`  
`constexpr iterator\_traits<_Iter>::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr iterator\_traits<_Iter>::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr pair<_Filter, _Filter> std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
`constexpr pair<_Filter, _Filter> std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`

- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp = typename iterator_traits<_OIter>::value_type>`  
`constexpr _OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _IIter, typename _Tp = typename iterator_traits<_IIter>::value_type>`  
`constexpr _IIter std::find (_IIter, _IIter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _IIter, typename _Predicate>`  
`constexpr _IIter std::find_if (_IIter, _IIter, _Predicate)`
- `template<typename _IIter, typename _Predicate>`  
`constexpr _IIter std::find_if_not (_IIter, _IIter, _Predicate)`
- `template<typename _IIter, typename _Funct>`  
`constexpr _Funct std::for_each (_IIter, _IIter, _Funct)`
- `template<typename _Filter, typename _Generator>`  
`constexpr void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`constexpr _OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _IIter1, typename _IIter2>`  
`constexpr bool std::includes (_IIter1, _IIter1, _IIter2, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _Compare>`  
`constexpr bool std::includes (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)`
- `template<typename _BIter>`  
`constexpr void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _Predicate>`  
`constexpr bool std::is_partitioned (_IIter, _IIter, _Predicate)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter>`  
`constexpr bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter)`

- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Ilter1, typename _Ilter2>`  
`constexpr bool std::lexicographical_compare (_Ilter1, _Ilter1, _Ilter2, _Ilter2)`
- `template<typename _Ilter1, typename _Ilter2, typename _Compare>`  
`constexpr bool std::lexicographical_compare (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Compare)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAlter>`  
`constexpr void std::make_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare>`  
`constexpr void std::make_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::max (initializer_list<_Tp>)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::max (initializer_list<_Tp>, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter>`  
`constexpr _Olter std::merge (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare>`  
`constexpr _Olter std::merge (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _Tp>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::min (initializer_list<_Tp>)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::min (initializer_list<_Tp>, _Compare)`
- `template<typename _Filter>`  
`constexpr _Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr _Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list<_Tp>)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list<_Tp>, _Compare)`



- `template<typename _Filter>`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare>`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _BIter>`  
`constexpr bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Iter, typename _Predicate>`  
`constexpr bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter>`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter>`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare>`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate>`  
`constexpr _BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate>`  
`constexpr pair< _OIter1, _OIter2 > std::partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter>`  
`constexpr void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter>`  
`constexpr bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare>`  
`constexpr bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator>`  
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr _Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp = typename iterator_traits<_Iter>::value_type>`  
`constexpr _OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`

- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`constexpr _OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`constexpr _Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp>`  
`constexpr _OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp = typename iterator_traits<_OIter>::value_type>`  
`constexpr _OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _BIter>`  
`constexpr void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter>`  
`constexpr _OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _Filter>`  
`constexpr _Filter std::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter>`  
`constexpr _OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate>`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _BinaryPredicate>`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator>`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter>`  
`constexpr void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter>`  
`constexpr void std::sort_heap (_RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate>`  
`constexpr _BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter>`  
`constexpr void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare>`  
`constexpr void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _Filter1, typename _Filter2>`  
`constexpr _Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`constexpr _OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation>`  
`constexpr _OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Filter>`  
`constexpr _Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate>`  
`constexpr _Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter>`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate>`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type>`  
`constexpr _Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp = typename iterator_traits<_Filter>::value_type, typename _Compare>`  
`constexpr _Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

### 7.18.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 7.19 parallel/algorithmfwd.h File Reference

### Namespaces

- namespace `std`
- namespace `std::__parallel`

### Functions

- `template<typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _Filter, typename _IterTag>`  
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter>`  
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _RAIter, typename _BiPredicate>`  
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`iterator\_traits<_Iter>::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`iterator\_traits<_RAIter>::difference_type std::__parallel::__count_if_switch (_RAIter __begin, _RAIter __↵  
end, _Predicate __pred, random\_access\_iterator\_tag, gnu\_parallel::\_\_Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`iterator_traits< _Iter >::difference_type std::__parallel::__count_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`iterator_traits< _RAIter >::difference_type std::__parallel::__count_switch ( _RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Iter std::__parallel::__find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2>`  
`_Iter std::__parallel::__find_first_of_switch ( _Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag>`  
`_RAIter std::__parallel::__find_first_of_switch ( _RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag>`  
`_Iter std::__parallel::__find_if_switch ( _Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::__parallel::__find_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag>`  
`_Iter std::__parallel::__find_switch ( _Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter std::__parallel::__find_switch ( _RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag>`  
`_Function std::__parallel::__for_each_switch ( _Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function std::__parallel::__for_each_switch ( _RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>`  
`_OIter std::__parallel::__generate_n_switch ( _OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter std::__parallel::__generate_n_switch ( _RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag>`  
`void std::__parallel::__generate_switch ( _Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator>`  
`void std::__parallel::__generate_switch ( _RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`bool std::__parallel::__lexicographical_compare_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::__parallel::__lexicographical_compare_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter std::__parallel::__max_element_switch ( _Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::__parallel::__max_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_OIter std::__parallel::__merge_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::__parallel::__merge_switch ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Filter, typename _Compare, typename _IterTag>`  
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair<_RAIter1, _RAIter2> std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag>`  
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag>`  
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &, _new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag>`  
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp>`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>`  
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>`  
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2>`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>`  
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_Olter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>`  
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2, typename _IterTag3>`  
`_Olter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, ↵`  
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename`  
`_IterTag3>`  
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, ↵`  
`_Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1,`  
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename`  
`_IterTag3>`  
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, ↵`  
`_IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>`  
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation>`  
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag,`  
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename ↵`  
`_Tag3>`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, ↵`  
`_Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation,`  
`random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism`  
`__parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>`  
`_RandomAccess_OIter std::parallel::unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, ↵`  
`_Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate>`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &↵`  
`value)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &↵`  
`value, __gnu_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filter>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate>`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each ( _Iter, _Iter, _Function)`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Generator>`  
`void std::__parallel::generate ( _Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::__parallel::generate_n ( _OIter, _Size, _Generator)`

- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator>`  
`_OIter std::parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter std::parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter>`  
`_Filter std::parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter>`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare>`  
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter>`  
`_Filter std::parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter>`  
`_Filter std::parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter>`  
`_Filter std::parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare>`  
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _↵`  
`Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _↵`  
`Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter std::__parallel::partition ( _Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate>`  
`_Filter std::__parallel::partition ( _Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle ( _RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle ( _RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace ( _Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace ( _Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Tp>`  
`void std::__parallel::replace ( _Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if ( _Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if ( _Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if ( _Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 std::__parallel::search ( _Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2>`  
`_Filter1 std::__parallel::search ( _Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`

- Generated by Doxygen

- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation>`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation>`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate>`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, __gnu_parallel::sequential_tag)`

### 7.19.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 7.20 align.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept`
- `template<size_t _Align, class _Tp>`  
`constexpr _Tp * std::assume\_aligned (_Tp * __ptr) noexcept`
- `template<size_t _Align, class _Tp>`  
`bool std::is\_sufficiently\_aligned (_Tp * __ptr)`

### 7.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.21 `bits/alloc_traits.h` File Reference

### Classes

- struct [std::allocator\\_traits<\\_Alloc>](#)
- struct [std::allocator\\_traits<allocator<\\_Tp>>](#)
- struct [std::allocator\\_traits<allocator<void>>](#)

### Namespaces

- namespace [std](#)

#### 7.21.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.22 `ext/alloc_traits.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits<\\_Alloc, typename>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.22.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.23 `allocated_ptr.h` File Reference

### Namespaces

- namespace [std](#)

#### 7.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.24 `allocator.h` File Reference

### Classes

- class [std::allocator<\\_Tp>](#)
- class [std::allocator<void>](#)

### Namespaces

- namespace [std](#)

#### 7.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.25 atomic\_base.h File Reference

### Classes

- struct [std::atomic\\_flag](#)

### Namespaces

- namespace [std](#)

### Macros

- #define `_GLIBCXX_ALWAYS_INLINE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_VAR_INIT(_V)`

### Typedefs

- typedef unsigned char `std::__atomic_flag_data_type`

### Enumerations

- enum class [std::memory\\_order](#) : int {  
    `relaxed` , `consume` , `acquire` , `release` ,  
    `acq_rel` , `seq_cst` }

### Functions

- void `std::atomic_signal_fence` ([memory\\_order](#) \_\_m) noexcept
- void `std::atomic_thread_fence` ([memory\\_order](#) \_\_m) noexcept
- template<typename \_Tp>  
  \_Tp `std::kill_dependency` (\_Tp \_\_y) noexcept
- constexpr [memory\\_order](#) `std::operator&` ([memory\\_order](#) \_\_m, \_\_memory\_order\_modifier \_\_mod) noexcept
- constexpr [memory\\_order](#) `std::operator|` ([memory\\_order](#) \_\_m, \_\_memory\_order\_modifier \_\_mod) noexcept

### Variables

- constexpr [memory\\_order](#) `std::memory_order_acq_rel`
- constexpr [memory\\_order](#) `std::memory_order_acquire`
- constexpr [memory\\_order](#) `std::memory_order_consume`
- constexpr [memory\\_order](#) `std::memory_order_relaxed`
- constexpr [memory\\_order](#) `std::memory_order_release`
- constexpr [memory\\_order](#) `std::memory_order_seq_cst`

#### 7.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 7.26 atomic\_futex.h File Reference

### Namespaces

- namespace [std](#)

### 7.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.27 `atomic_lockfree_defines.h` File Reference

### Macros

- `#define` [ATOMIC\\_BOOL\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR16\\_T\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR32\\_T\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_CHAR\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_INT\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_LLONG\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_LONG\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_POINTER\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_SHORT\\_LOCK\\_FREE](#)
- `#define` [ATOMIC\\_WCHAR\\_T\\_LOCK\\_FREE](#)

### 7.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 7.28 `atomic_timed_wait.h` File Reference

### 7.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 7.29 `atomic_wait.h` File Reference

### 7.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 7.30 `basic_ios.h` File Reference

### Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits>](#)

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Facet>`  
`const _Facet & std::\_\_check\_facet (const _Facet * __f)`

### 7.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 7.31 `basic_ios.tcc` File Reference

### Namespaces

- namespace [std](#)

#### 7.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 7.32 `basic_string.h` File Reference

### Classes

- class [std::basic\\_string<\\_CharT, \\_Traits, \\_Alloc>](#)
- struct [std::hash<basic\\_string<char, char\\_traits<char>, \\_Alloc>>](#)
- struct [std::hash<basic\\_string<char16\\_t, char\\_traits<char16\\_t>, \\_Alloc>>](#)
- struct [std::hash<basic\\_string<char32\\_t, char\\_traits<char32\\_t>, \\_Alloc>>](#)
- struct [std::hash<basic\\_string<wchar\\_t, char\\_traits<wchar\\_t>, \\_Alloc>>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define _GLIBCXX_TO_STRING_CONSTEXPR`

### Functions

- `template<typename _Str>`  
`constexpr _Str std::__str_concat (typename _Str::value_type const * __lhs, typename _Str::size_type __lhs_↵`  
`len, typename _Str::value_type const * __rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type`  
`const & __a)`
- `constexpr void std::__to_wstring_numeric (const char * __s, int __len, wchar_t * __wout)`
- `wstring std::__to_wstring_numeric (const string & __s)`
- `template<typename _InputIterator, typename _CharT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator =`  
`allocator<_CharT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::basic_string (_InputIterator, _InputIterator, _Allocator= _Allocator()) -> basic_string<_CharT, char\_traits<`  
`_CharT>, _Allocator>`
- `template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>`  
`std::basic_string (basic\_string\_view<_CharT, _Traits>, const _Allocator &= _Allocator()) -> basic_string<↵`  
`_CharT, _Traits, _Allocator>`
- `template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>`  
`std::basic_string (basic\_string\_view<_CharT, _Traits>, typename basic_string<_CharT, _Traits, _Allocator`  
`>::size_type, typename basic_string<_CharT, _Traits, _Allocator>::size_type, const _Allocator &= _Allocator())`  
`-> basic_string<_CharT, _Traits, _Allocator>`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_istream<_CharT, _Traits> & std::getline (basic\_istream<_CharT, _Traits> && __is, basic\_string<↵`  
`_CharT, _Traits, _Alloc> & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic\_istream<_CharT, _Traits> & std::getline (basic\_istream<_CharT, _Traits> && __is, basic\_string<↵`  
`_CharT, _Traits, _Alloc> & __str, _CharT __delim)`

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<> basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<> basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ ( _CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ ( _CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT * __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits > (0))`



- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<> basic_istream< char > & std::operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `double std::stod (const string &__str, size_t * __idx=0)`
- `double std::stod (const wstring &__str, size_t * __idx=0)`
- `float std::stof (const string &__str, size_t * __idx=0)`
- `float std::stof (const wstring &__str, size_t * __idx=0)`
- `int std::stoi (const string &__str, size_t * __idx=0, int __base=10)`
- `int std::stoi (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t * __idx=0, int __base=10)`
- `long std::stol (const wstring &__str, size_t * __idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t * __idx=0)`
- `long double std::stold (const wstring &__str, size_t * __idx=0)`
- `long long std::stoll (const string &__str, size_t * __idx=0, int __base=10)`
- `long long std::stoll (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long std::stoul (const wstring &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t * __idx=0, int __base=10)`
- `unsigned long long std::stoull (const wstring &__str, size_t * __idx=0, int __base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`constexpr void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept( /*conditional */)`
- `string std::to_string (double __val)`
- `string std::to_string (float __val)`
- `string std::to_string (int __val) noexcept`
- `string std::to_string (long __val) noexcept`
- `string std::to_string (long double __val)`
- `string std::to_string (long long __val)`
- `string std::to_string (unsigned __val) noexcept`
- `string std::to_string (unsigned long __val) noexcept`
- `string std::to_string (unsigned long long __val)`
- `wstring std::to_wstring (double __val)`
- `wstring std::to_wstring (float __val)`
- `wstring std::to_wstring (int __val)`
- `wstring std::to_wstring (long __val)`
- `wstring std::to_wstring (long double __val)`
- `wstring std::to_wstring (long long __val)`
- `wstring std::to_wstring (unsigned __val)`
- `wstring std::to_wstring (unsigned long __val)`
- `wstring std::to_wstring (unsigned long long __val)`

### 7.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 7.33 `basic_string.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc>  
basic\_istream< _CharT, _Traits > & std::getline (basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>  
basic\_istream< _CharT, _Traits > & std::operator>> (basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`

### 7.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 7.34 `backward/binders.h` File Reference

### Classes

- class [std::binder1st](#)< [\\_Operation](#) >
- class [std::binder2nd](#)< [\\_Operation](#) >

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Operation, typename _Tp>  
binder1st< \_Operation > std::bind1st (const \_Operation &__fn, const \_Tp &__x)`
- `template<typename _Operation, typename _Tp>  
binder2nd< \_Operation > std::bind2nd (const \_Operation &__fn, const \_Tp &__x)`

### 7.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.35 `bits/binders.h` File Reference

### Namespaces

- namespace [std](#)

## Typedefs

- `template<typename _Fn, typename... _Args>`  
using **std::\_Bind\_back\_t**
- `template<typename _Fn, typename... _Args>`  
using **std::\_Bind\_front\_t**

## Functions

- `template<typename... _BoundArgs, typename... _Args>`  
constexpr auto **std::\_make\_bound\_args** (\_Args &&... \_\_args)

### 7.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.36 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

### Functions

- `template<class _Tp>`  
void **\_\_gnu\_cxx::\_\_aux\_require\_boolean\_expr** (const \_Tp &\_\_t)
- void **\_\_gnu\_cxx::\_\_error\_type\_must\_be\_a\_signed\_integer\_type** ()
- void **\_\_gnu\_cxx::\_\_error\_type\_must\_be\_an\_integer\_type** ()
- void **\_\_gnu\_cxx::\_\_error\_type\_must\_be\_an\_unsigned\_integer\_type** ()
- `template<class _Concept>`  
constexpr void **\_\_gnu\_cxx::\_\_function\_requires** ()

### 7.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.37 c++0x\_warning.h File Reference

### 7.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.38 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)
- struct [std::char\\_traits<\\_CharT>](#)
- struct [std::char\\_traits<char>](#)
- struct [std::char\\_traits<wchar\\_t>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define _GLIBCXX_ALWAYS_INLINE`

### Functions

- `template<typename _ChTraits>  
constexpr auto std::\_\_detail::\_\_char\_traits\_cmp\_cat (int __cmp) noexcept`

#### 7.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 7.39 charconv.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _Tp>  
constexpr void std::\_\_detail::\_\_to\_chars\_10\_impl (char *__first, unsigned __len, _Tp __val) noexcept`
- `template<typename _Tp>  
constexpr unsigned std::\_\_detail::\_\_to\_chars\_len (_Tp __value, int __base) noexcept`

#### 7.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<charconv>`.

## 7.40 chrono.h File Reference

### Classes

- struct [std::common\\_type<chrono::duration<\\_Rep, \\_Period>>](#)
- struct [std::common\\_type<chrono::duration<\\_Rep, \\_Period>, chrono::duration<\\_Rep, \\_Period>>](#)
- struct [std::common\\_type<chrono::duration<\\_Rep1, \\_Period1>, chrono::duration<\\_Rep2, \\_Period2>>](#)

- struct `std::common_type< chrono::time_point< _Clock, _Duration > >`
- struct `std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >`
- struct `std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >`
- class `std::chrono::duration< _Rep, _Period >`
- struct `std::chrono::duration_values< _Rep >`
- struct `std::chrono::steady_clock`
- struct `std::chrono::system_clock`
- class `std::chrono::time_point< _Clock, _Dur >`
- struct `std::chrono::treat_as_floating_point< _Rep >`

## Namespaces

- namespace `std`
- namespace `std::chrono`
- namespace `std::filesystem`
- namespace `std::literals`
- namespace `std::literals::chrono_literals`

## Typedefs

- using `std::chrono::days`
- using `std::chrono::file_clock`
- template<typename \_Duration>  
using `std::chrono::file_time`
- using `std::chrono::high_resolution_clock`
- using `std::chrono::hours`
- using `std::chrono::microseconds`
- using `std::chrono::milliseconds`
- using `std::chrono::minutes`
- using `std::chrono::months`
- using `std::chrono::nanoseconds`
- using `std::chrono::seconds`
- using `std::chrono::sys_days`
- using `std::chrono::sys_seconds`
- template<typename \_Duration>  
using `std::chrono::sys_time`
- using `std::chrono::weeks`
- using `std::chrono::years`

## Functions

- template<typename \_Rep, typename \_Period>  
constexpr `enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > >` `std::chrono::abs`  
(`duration< _Rep, _Period > __d`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::__detail::ceil` (`const duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Rep, typename \_Period>  
constexpr `__enable_if_is_duration< _ToDur >` `std::chrono::ceil` (`const duration< _Rep, _Period > &__d`)
- template<typename \_ToDur, typename \_Clock, typename \_Dur>  
constexpr `enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > >` `std::chrono::ceil` (`const time_point< _Clock, _Dur > &__tp`)

- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)`
- `template<char... _Digits>`  
`constexpr chrono::hours std::literals::chrono_literals::operator""h ()`
- `constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long double __hours)`
- `template<char... _Digits>`  
`constexpr chrono::minutes std::literals::chrono_literals::operator""min ()`
- `constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)`
- `template<char... _Digits>`  
`constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()`
- `constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __msecs)`
- `template<char... _Digits>`  
`constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()`
- `constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)`
- `template<char... _Digits>`  
`constexpr chrono::seconds std::literals::chrono_literals::operator""s ()`
- `constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)`
- `template<char... _Digits>`  
`constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()`
- `constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __usecs)`
- `template<typename _ToDur, typename _Rep, typename _Period>`  
`constexpr enable_if_t< __and< __is_duration< _ToDur >, __not< treat_as_floating_point< typename _ToDur::rep > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)`
- `template<typename _ToDur, typename _Clock, typename _Dur>`  
`constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`

- `template<typename _Rep1, typename _Rep2, typename _Period>`  
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s,`  
`const duration< _Rep2, _Period > &__d)`
- `template<typename _Rep1, typename _Period, typename _Rep2>`  
`constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period >`  
`std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 >`  
`&__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr auto std::chrono::operator<= > (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>`  
`constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2,`  
`_Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>`  
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >`  
`std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>`  
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >`  
`std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__`  
`__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__`  
`__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator< (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator<= (const time_point< _Clock, _Dur1 > &__lhs, const time_point< __`  
`__Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>`  
`constexpr auto std::chrono::operator<= > (const time_point< _Clock, _Dur1 > &__lhs, const time_point< __`  
`__Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator> (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,`  
`_Dur2 > &__rhs)`

- `template<typename _Clock, typename _Dur1, typename _Dur2>`  
`constexpr bool std::chrono::operator>= (const time\_point< _Clock, _Dur1 > &__lhs, const time\_point< _↔`  
`Clock, _Dur2 > &__rhs)`

## Variables

- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v`
- `template<typename _Tp>`  
`constexpr bool std::chrono::is_clock_v< _Tp >`
- `template<> constexpr bool std::chrono::is_clock_v< file\_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< steady\_clock >`
- `template<> constexpr bool std::chrono::is_clock_v< system\_clock >`
- `template<typename _Rep>`  
`constexpr bool std::chrono::treat_as_floating_point_v`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< float >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< int >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long double >`
- `template<> constexpr bool std::chrono::treat_as_floating_point_v< long long >`

### 7.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

### 7.40.2 Function Documentation

#### `ceil()`

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
 const duration< _Rep, _Period > & __d) [nodiscard], [constexpr]
```

Convert a duration to type `ToDur` and round up.

If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

#### Template Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>_ToDur</code> | The result type must be a duration. |
|---------------------|-------------------------------------|

#### Parameters

|                  |             |
|------------------|-------------|
| <code>__d</code> | A duration. |
|------------------|-------------|

#### Returns

The value of `__d` converted to type `_ToDur`.



Since

C++17

## 7.41 chrono\_io.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Functions

- void **std::chrono::\_\_detail::from\_stream** ()=delete
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, day &\_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Rep, typename \_Period, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, [duration](#)< \_Rep, \_Period > &\_\_d, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, file\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, gps\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, local\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, month &\_\_m, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, month\_day &\_\_md, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, sys\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, tai\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)
- template<typename \_CharT, typename \_Traits, typename \_Duration, typename \_Alloc = allocator<\_CharT>>  
[basic\\_istream](#)< \_CharT, \_Traits > & **std::chrono::from\_stream** ([basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, const \_CharT \*\_\_fmt, utc\_time< \_Duration > &\_\_tp, [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > \*\_\_abbrev=nullptr, [minutes](#) \*\_\_offset=nullptr)

- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, weekday & __wd, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year & __y, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year_month & __ym, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes * __`  
`offset=nullptr)`
- `template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>`  
`basic_istream< _CharT, _Traits > & std::chrono::from_stream (basic_istream< _CharT, _Traits > &__is, const`  
`_CharT * __fmt, year_month_day & __ymd, basic_string< _CharT, _Traits, _Alloc > * __abbrev=nullptr, minutes`  
`* __offset=nullptr)`
- `template<typename _Duration>`  
`__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const`  
`string * __abbrev=nullptr, const seconds * __offset_sec=nullptr)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const day & __d)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const file_time< _Duration > & __t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const gps_time< _Duration > & __t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const hh_mm_ss< _Duration > & __hms)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`local_info & __li)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`requires requires(const sys_time<_Duration>& __st) { __os << __st; }`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const local_time< _Duration > & __lt)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const month & __m)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const month_day & __md)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const month_day_last & __mdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const month_weekday & __mwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const month_weekday_last & __mwdl)`

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const sys_days &__dp)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const`  
`sys_info &__i)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const sys_time< _Duration > &__tp)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const tai_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits, typename _Duration>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const utc_time< _Duration > &__t)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday &__wd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_indexed &__wdi)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const weekday_last &__wdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year &__y)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month &__ym)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day &__ymd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_day_last &__ymdl)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday &__ymwd)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const year_month_weekday_last &__ymwdl)`
- `template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os,`  
`const zoned_time< _Duration, _TimeZonePtr > &__t)`
- `template<typename _CharT, typename _Traits, typename _Rep, typename _Period>`  
`basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const duration< _Rep, _Period > &__d)`
- `template<typename _CharT, __detail::__parsable< _CharT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string< _CharT, _Traits, _Alloc>, __detail::__↵`  
`↵parsable< _CharT, _Traits, _StrT > _Parsable>`  
`auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string< _CharT, _Traits, _Alloc > & __↵`  
`↵_abbrev)`

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵  
parsable<_CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, basic_string<_CharT, _Traits, _Alloc > & __↵  
_abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__↵  
parsable<_CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const _CharT * __fmt, _Parsable & __tp, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable<_CharT, _Traits> _Parsable>  
auto std::chrono::parse (const basic_string<_CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵  
parsable<_CharT, _Traits, _StrT > _Parsable>  
auto std::chrono::parse (const basic_string<_CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string<↵  
_CharT, _Traits, _Alloc > & __abbrev)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↵  
parsable<_CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const basic_string<_CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, basic_string<↵  
_CharT, _Traits, _Alloc > & __abbrev, minutes & __offset)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<↵  
_CharT, _Traits, _StrT, minutes > _Parsable>  
auto std::chrono::parse (const basic_string<_CharT, _Traits, _Alloc > & __fmt, _Parsable & __tp, minutes & __↵  
_offset)`

#### 7.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

## 7.42 codecvt.h File Reference

### Classes

- class `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >`
- class `std::codecvt<_InternT, _ExternT, _StateT >`
- class `std::codecvt<char, char, mbstate_t >`
- class `std::codecvt<char16_t, char, mbstate_t >`
- class `std::codecvt<char32_t, char, mbstate_t >`
- class `std::codecvt<wchar_t, char, mbstate_t >`
- class `std::codecvt_base`
- class `std::codecvt_byname<_InternT, _ExternT, _StateT >`

### Namespaces

- namespace `std`

### Functions

- `template bool std::has_facet< codecvt<char, char, mbstate_t > > (const locale &)`
- `template bool std::has_facet< codecvt<wchar_t, char, mbstate_t > > (const locale &)`
- `template const codecvt<char, char, mbstate_t > & std::use_facet< codecvt<char, char, mbstate_t > > (const locale &)`
- `template const codecvt<wchar_t, char, mbstate_t > & std::use_facet< codecvt<wchar_t, char, mbstate_t > > (const locale &)`

## Variables

- `template<typename _InternT, typename _ExternT, typename _StateT>`  
`locale::id std::codecvt< _InternT, _ExternT, _StateT >::id`

### 7.42.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.43 concept\_check.h File Reference

### Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

### 7.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.44 cow\_string.h File Reference

### Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Operation>`  
`void std::basic_string< _CharT, _Traits, _Alloc >__resize_and_overwrite (const size\_type __n, _Operation __op)`

### Variables

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`const basic\_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::npos`

### 7.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

Defines the reference-counted COW string implementation.

## 7.45 cpp\_type\_traits.h File Reference

### Namespaces

- namespace `std`

## Macros

- `#define __INT_N(TYPE)`

## Functions

- `template<typename _Iterator>`  
`constexpr _Iterator std::__miter_base (_Iterator __it)`

## Variables

- `template<typename _ValT, typename _Tp>`  
`constexpr bool std::__can_use_memchr_for_find`

### 7.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits.h>`.

## 7.46 cpyfunc\_impl.h File Reference

### Classes

- class `std::copyable\_function< \_Res(\_ArgTypes...) \_GLIBCXX\_MOF\_CV noexcept(\_Noex)>`

### Namespaces

- namespace `std`

## Macros

- `#define _GLIBCXX_MOF_CV_REF`
- `#define _GLIBCXX_MOF_INV_QUALS`
- `#define _GLIBCXX_MOF_REF`

### 7.46.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.47 cxxabi\_forced.h File Reference

### Classes

- class `\_\_cxxabiv1::\_\_forced\_unwind`

### 7.47.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 7.48 cxxabi\_init\_exception.h File Reference

### Namespaces

- namespace `std`

## Macros

- `#define _GLIBCXX_CDTOR_CALLABI`
- `#define _GLIBCXX_HAVE_CDTOR_CALLABI`

## Functions

- `void * __cxxabiv1::__cxa_allocate_exception (size_t) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `__cxa_refcounted_exception * __cxxabiv1::__cxa_init_primary_exception (void * __object, std::type\_info * __tinfo, void (* __dest)(void *)) noexcept`

### 7.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.49 deque.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>  
::Deque\_iterator< _OTp, _OTp &, _OTp * > std::copy\_move\_a1 (::Deque\_iterator< _ITp, _IRef, _IPtr > __first, ::Deque\_iterator< _ITp, _IRef, _IPtr > __last, ::Deque\_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy\_move\_a1 (::Deque\_iterator< _Tp, _Ref, _Ptr > __first, ::Deque\_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>  
__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, ::Deque\_iterator< _Tp, _Tp &, _Tp * > >::__type std::copy\_move\_a1 (_II __first, _II __last, ::Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _CharT>  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::Deque\_iterator< _CharT, _CharT &, _CharT * > >::__type std::copy\_move\_a2 (istreambuf\_iterator< _CharT, char\_traits< _CharT > > __first, istreambuf\_iterator< _CharT, char\_traits< _CharT > > __last, ::Deque\_iterator< _CharT, _CharT &, \_\_CharT * > __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>  
::Deque\_iterator< _OTp, _OTp &, _OTp * > std::copy\_move\_backward\_a1 (::Deque\_iterator< _ITp, \_\_IRef, _IPtr > __first, ::Deque\_iterator< _ITp, _IRef, _IPtr > __last, ::Deque\_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy\_move\_backward\_a1 (::Deque\_iterator< _Tp, _Ref, _Ptr > __first, ::Deque\_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>  
__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, ::Deque\_iterator< _Tp, _Tp &, _Tp * > >::__type std::copy\_move\_backward\_a1 (_II __first, _II __last, ::Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy\_move\_backward\_dit (::Deque\_iterator< _Tp, _Ref, _Ptr > __first, ::Deque\_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>  
_OI std::copy\_move\_dit (::Deque\_iterator< _Tp, _Ref, _Ptr > __first, ::Deque\_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`

- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::_Deque_iterator< _CharT, _CharT &, _CharT *`  
`> >::__type std::__copy_n_a (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size,`  
`::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>`  
`__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1`  
`(::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__equal_aux1 (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1,`  
`_Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr>`  
`__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1`  
`(_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II>`  
`bool std::__equal_dit (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp,`  
`_Ref, _Ptr > &__last1, _II __first2)`
- `template<typename _Tp, typename _VTP>`  
`void std::__fill_a1 (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > &__last, const _VTP &__value)`
- `template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>`  
`int std::__lex_cmp_dit (::_Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp1, _Ref, _Ptr >`  
`__last1, const _Tp2 *__first2, const _Tp2 *__last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator<`  
`_Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, ↵`  
`_Ref2, _Ptr2 > __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool std::__lexicographical_compare_aux1 (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator<`  
`_Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2,`  
`_Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>`  
`::_Deque_iterator< _OTp, _OTp &, _OTp * > std::__uninitialized_copy_a (::_Deque_iterator< _ITp, _IRef, ↵`  
`_IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * >`  
`__result, allocator< _Tp > &)`
- `template<typename _Iter, typename _OTp, typename _Tp>`  
`__enable_if_t< __is_random_access_iter< _Iter >::__value, ::_Deque_iterator< _OTp, _OTp &, _OTp * > >`  
`std::__uninitialized_copy_a (_Iter __first, _Iter __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result,`  
`allocator< _Tp > &)`
- `template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>`  
`::_Deque_iterator< _OTp, _OTp &, _OTp * > std::__uninitialized_move_a (::_Deque_iterator< _ITp, _IRef,`  
`_IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * >`  
`__result, allocator< _Tp > &)`

### 7.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

## 7.50 enable\_special\_members.h File Reference

### Namespaces

- namespace `std`



### 7.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.51 erase\_if.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>  
constexpr _Container::size_type std::__detail::__erase_if (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)`
- `template<typename _Container, typename _UnsafeContainer, typename _Predicate>  
_Container::size_type std::__detail::__erase_nodes_if (_Container &__cont, _UnsafeContainer &__ucont, _↵  
Predicate __pred)`

### 7.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.52 exception.h File Reference

### Classes

- class [std::exception](#)

### Namespaces

- namespace [std](#)

### 7.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.53 exception\_defines.h File Reference

### Macros

- `#define __catch(X)`
- `#define __throw_exception_again`
- `#define __try`

### 7.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 7.54 exception\_ptr.h File Reference

### Classes

- class [std::\\_\\_unspecified\\_\\_::exception\\_ptr](#)
- class [std::exception\\_ptr](#)

## Namespaces

- namespace [std](#)

## Functions

- [exception\\_ptr](#) [std::current\\_exception](#) () noexcept
- [template](#)<typename [\\_Ex](#)>  
constexpr [exception\\_ptr](#) [std::make\\_exception\\_ptr](#) ([\\_Ex](#) \_\_ex) noexcept
- void [std::\\_\\_unspecified\\_\\_::rethrow\\_exception](#) ([exception\\_ptr](#))
- void [std::rethrow\\_exception](#) ([exception\\_ptr](#))
- constexpr void [std::swap](#) ([exception\\_ptr](#) &\_\_lhs, [exception\\_ptr](#) &\_\_rhs)

### 7.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

### 7.54.2 Function Documentation

#### [rethrow\\_exception\(\)](#)

```
void std::rethrow_exception (
 exception_ptr)
```

Throw the object pointed to by the [exception\\_ptr](#).

## 7.55 [formatfwd.h](#) File Reference

### 7.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.

## 7.56 [forward\\_list.h](#) File Reference

### Classes

- struct [std::\\_Fwd\\_list\\_base](#)< [\\_Tp](#), [\\_Alloc](#) >
- struct [std::\\_Fwd\\_list\\_const\\_iterator](#)< [\\_Tp](#) >
- struct [std::\\_Fwd\\_list\\_iterator](#)< [\\_Tp](#) >
- struct [std::\\_Fwd\\_list\\_node](#)< [\\_Tp](#) >
- struct [std::\\_Fwd\\_list\\_node\\_base](#)
- class [std::\\_\\_fwdlst::\\_Iterator](#)< [\\_Const](#), [\\_Ptr](#) >
- struct [std::\\_\\_fwdlst::\\_Node](#)< [\\_ValPtr](#) >
- struct [std::\\_\\_fwdlst::\\_Node\\_base](#)< [\\_VoidPtr](#) >
- class [std::forward\\_list](#)< [\\_Tp](#), [\\_Alloc](#) >

## Namespaces

- namespace [std](#)

## Macros

- [#define](#) [\\_GLIBCXX\\_FWDLIST\\_REMOVE\\_RETURN\\_TYPE\\_TAG](#)
- [#define](#) [\\_GLIBCXX\\_USE\\_ALLOC\\_PTR\\_FOR\\_FWD\\_LIST](#)

## Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::forward_list(_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Alloc>`  
`__detail::__synth3way_t<_Tp> std::operator<=> (const forward_list<_Tp, _Alloc> &__x, const forward_list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const forward_list<_Tp, _Alloc> &__lx, const forward_list<_Tp, _Alloc> &__ly)`
- `template<typename _Tp, typename _Alloc>`  
`void std::swap (forward_list<_Tp, _Alloc> &__lx, forward_list<_Tp, _Alloc> &__ly) noexcept(noexcept(__lx.swap(__ly)))`

### 7.56.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 7.57 forward\_list.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX20_ONLY(__expr)`

## Functions

- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const forward_list<_Tp, _Alloc> &__lx, const forward_list<_Tp, _Alloc> &__ly)`

### 7.57.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 7.58 bits/fs\_dir.h File Reference

### Classes

- class `std::filesystem::directory_entry`
- class `std::filesystem::directory_iterator`
- class `std::filesystem::file_status`
- class `std::filesystem::recursive_directory_iterator`

### Namespaces

- namespace `std`
- namespace `std::filesystem`

## Variables

- `template<> constexpr bool std::ranges::enable_borrowed_range< filesystem::directory_iterator >`
- `template<> constexpr bool std::ranges::enable_borrowed_range< filesystem::recursive_directory_iterator >`
- `template<> constexpr bool std::ranges::enable_view< filesystem::directory_iterator >`
- `template<> constexpr bool std::ranges::enable_view< filesystem::recursive_directory_iterator >`

### 7.58.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 7.59 experimental/bits/fs\_dir.h File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Functions

- `directory_iterator std::experimental::filesystem::begin (directory_iterator __iter) noexcept`
- `recursive_directory_iterator std::experimental::filesystem::begin (recursive_directory_iterator __iter) noexcept`
- `directory_iterator std::experimental::filesystem::end (directory_iterator) noexcept`
- `recursive_directory_iterator std::experimental::filesystem::end (recursive_directory_iterator) noexcept`
- `bool std::experimental::filesystem::operator!= (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator!= (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator== (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::operator== (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`

### 7.59.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 7.60 bits/fs\_fwd.h File Reference

### Classes

- struct `std::filesystem::space_info`

### Namespaces

- namespace `std`
- namespace `std::filesystem`

### Typedefs

- using `std::filesystem::file_time_type`

## Enumerations

- enum class `std::filesystem::copy_options` : unsigned short { `none` , `skip_existing` , `overwrite_existing` , `update_existing` , `recursive` , `copy_symlinks` , `skip_symlinks` , `directories_only` , `create_symlinks` , `create_hard_links` }
- enum class `std::filesystem::directory_options` : unsigned char { `none` , `follow_directory_symlink` , `skip_permission_denied` }
- enum class `std::filesystem::file_type` : signed char { `none` , `not_found` , `regular` , `directory` , `symlink` , `block` , `character` , `fifo` , `socket` , `unknown` }
- enum class `std::filesystem::perm_options` : unsigned { `replace` , `add` , `remove` , `nofollow` }
- enum class `std::filesystem::perms` : unsigned { `none` , `owner_read` , `owner_write` , `owner_exec` , `owner_all` , `group_read` , `group_write` , `group_exec` , `group_all` , `others_read` , `others_write` , `others_exec` , `others_all` , `all` , `set_uid` , `set_gid` , `sticky_bit` , `mask` , `unknown` }

## Functions

- void `std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options)
- void `std::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options, `error_code` &)
- bool `std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option)
- bool `std::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_option, `error_code` &)
- `path` `std::filesystem::current_path` ()
- bool `std::filesystem::exists` (`file_status`) noexcept
- uintmax\_t `std::filesystem::file_size` (const `path` &)
- uintmax\_t `std::filesystem::file_size` (const `path` &, `error_code` &) noexcept
- uintmax\_t `std::filesystem::hard_link_count` (const `path` &)
- uintmax\_t `std::filesystem::hard_link_count` (const `path` &, `error_code` &) noexcept
- bool `std::filesystem::is_other` (`file_status`) noexcept
- bool `std::filesystem::is_regular_file` (`file_status`) noexcept
- bool `std::filesystem::is_symlink` (`file_status`) noexcept
- `file_time_type` `std::filesystem::last_write_time` (const `path` &)
- `file_time_type` `std::filesystem::last_write_time` (const `path` &, `error_code` &) noexcept
- `copy_options` & `std::filesystem::operator&=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
- constexpr `copy_options` `std::filesystem::operator^=` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
- `copy_options` & `std::filesystem::operator^=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
- constexpr `copy_options` `std::filesystem::operator|` (`copy_options` \_\_x, `copy_options` \_\_y) noexcept
- `copy_options` & `std::filesystem::operator|=` (`copy_options` &\_\_x, `copy_options` \_\_y) noexcept
- constexpr `copy_options` `std::filesystem::operator~` (`copy_options` \_\_x) noexcept
- void `std::filesystem::permissions` (const `path` &, `perms`, `perm_options`, `error_code` &) noexcept
- `path` `std::filesystem::proximate` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path` `std::filesystem::relative` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- bool `std::filesystem::remove` (const `path` &, `error_code` &) noexcept
- uintmax\_t `std::filesystem::remove_all` (const `path` &)
- uintmax\_t `std::filesystem::remove_all` (const `path` &, `error_code` &)
- `file_status` `std::filesystem::status` (const `path` &)
- `file_status` `std::filesystem::status` (const `path` &, `error_code` &) noexcept
- bool `std::filesystem::status_known` (`file_status`) noexcept

- `file_status std::filesystem::symlink_status (const path &)`
- `file_status std::filesystem::symlink_status (const path &, error_code &) noexcept`
- `perms & std::filesystem::operator&= (perms &__x, perms __y) noexcept`
- `constexpr perms std::filesystem::operator^ (perms __x, perms __y) noexcept`
- `perms & std::filesystem::operator^= (perms &__x, perms __y) noexcept`
- `constexpr perms std::filesystem::operator| (perms __x, perms __y) noexcept`
- `perms & std::filesystem::operator|= (perms &__x, perms __y) noexcept`
- `constexpr perms std::filesystem::operator~ (perms __x) noexcept`
- `perm_options & std::filesystem::operator&= (perm_options &__x, perm_options __y) noexcept`
- `constexpr perm_options std::filesystem::operator^ (perm_options __x, perm_options __y) noexcept`
- `perm_options & std::filesystem::operator^= (perm_options &__x, perm_options __y) noexcept`
- `constexpr perm_options std::filesystem::operator| (perm_options __x, perm_options __y) noexcept`
- `perm_options & std::filesystem::operator|= (perm_options &__x, perm_options __y) noexcept`
- `constexpr perm_options std::filesystem::operator~ (perm_options __x) noexcept`
- `directory_options & std::filesystem::operator&= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator^ (directory_options __x, directory_options __y) noexcept`
- `directory_options & std::filesystem::operator^= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator| (directory_options __x, directory_options __y) noexcept`
- `directory_options & std::filesystem::operator|= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::filesystem::operator~ (directory_options __x) noexcept`

### 7.60.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 7.61 experimental/bits/fs\_fwd.h File Reference

### Classes

- struct `std::experimental::filesystem::v1::space_info`

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Typedefs

- using `std::experimental::filesystem::file_time_type`

## Enumerations

- enum class `std::experimental::filesystem::copy_options` : unsigned short { `none`, `skip_existing`, `overwrite_existing`, `update_existing`, `recursive`, `copy_symlinks`, `skip_symlinks`, `directories_only`, `create_symlinks`, `create_hard_links` }
- enum class `std::experimental::filesystem::directory_options` : unsigned char { `none`, `follow_directory_symlink`, `skip_permission_denied` }
- enum class `std::experimental::filesystem::file_type` : signed char { `none`, `not_found`, `regular`, `directory`, `symlink`, `block`, `character`, `fifo`, `socket`, `unknown` }
- enum class `std::experimental::filesystem::perms` : unsigned { `none`, `owner_read`, `owner_write`, `owner_exec`, `owner_all`, `group_read`, `group_write`, `group_exec`, `group_all`, `others_read`, `others_write`, `others_exec`, `others_all`, `all`, `set_uid`, `set_gid`, `sticky_bit`, `mask`, `unknown`, `add_perms`, `remove_perms`, `symlink_nofollow` }

## Functions

- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_options)
- void `std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_options, error\_code &) noexcept
- bool `std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_option)
- bool `std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to, copy\_options \_\_option, error\_code &)
- `path` `std::experimental::filesystem::current_path` ()
- bool `std::experimental::filesystem::is_regular_file` (file\_status) noexcept
- bool `std::experimental::filesystem::is_symlink` (file\_status) noexcept
- copy\_options & `std::experimental::filesystem::operator&=` (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- constexpr copy\_options `std::experimental::filesystem::operator^` (copy\_options \_\_x, copy\_options \_\_y) noexcept
- copy\_options & `std::experimental::filesystem::operator^=` (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- constexpr copy\_options `std::experimental::filesystem::operator|` (copy\_options \_\_x, copy\_options \_\_y) noexcept
- copy\_options & `std::experimental::filesystem::operator|=` (copy\_options &\_\_x, copy\_options \_\_y) noexcept
- constexpr copy\_options `std::experimental::filesystem::operator~` (copy\_options \_\_x) noexcept
- file\_status `std::experimental::filesystem::status` (const `path` &)
- file\_status `std::experimental::filesystem::status` (const `path` &, error\_code &) noexcept
- bool `std::experimental::filesystem::status_known` (file\_status) noexcept
- file\_status `std::experimental::filesystem::symlink_status` (const `path` &)
- file\_status `std::experimental::filesystem::symlink_status` (const `path` &, error\_code &) noexcept
- perms & `std::experimental::filesystem::operator&=` (perms &\_\_x, perms \_\_y) noexcept
- constexpr perms `std::experimental::filesystem::operator^` (perms \_\_x, perms \_\_y) noexcept
- perms & `std::experimental::filesystem::operator^=` (perms &\_\_x, perms \_\_y) noexcept
- constexpr perms `std::experimental::filesystem::operator|` (perms \_\_x, perms \_\_y) noexcept
- perms & `std::experimental::filesystem::operator|=` (perms &\_\_x, perms \_\_y) noexcept
- constexpr perms `std::experimental::filesystem::operator~` (perms \_\_x) noexcept

- `directory_options & std::experimental::filesystem::operator&= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator^ (directory_options __x, directory_options __y) noexcept`
- `directory_options & std::experimental::filesystem::operator^= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator| (directory_options __x, directory_options __y) noexcept`
- `directory_options & std::experimental::filesystem::operator|= (directory_options &__x, directory_options __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::operator~ (directory_options __x) noexcept`

### 7.61.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 7.62 bits/fs\_ops.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::filesystem](#)

### Functions

- [path](#) `std::filesystem::absolute (const path &__p)`
- [path](#) `std::filesystem::absolute (const path &__p, error_code &__ec)`
- [path](#) `std::filesystem::canonical (const path &__p)`
- [path](#) `std::filesystem::canonical (const path &__p, error_code &__ec)`
- void `std::filesystem::copy (const path &__from, const path &__to)`
- void `std::filesystem::copy (const path &__from, const path &__to, copy_options __options)`
- void `std::filesystem::copy (const path &__from, const path &__to, copy_options __options, error_code &__ec)`
- void `std::filesystem::copy (const path &__from, const path &__to, error_code &__ec)`
- bool `std::filesystem::copy_file (const path &__from, const path &__to)`
- bool `std::filesystem::copy_file (const path &__from, const path &__to, copy_options __option)`
- bool `std::filesystem::copy_file (const path &__from, const path &__to, copy_options __option, error_code &__ec)`
- bool `std::filesystem::copy_file (const path &__from, const path &__to, error_code &__ec)`
- void `std::filesystem::copy_symlink (const path &__existing_symlink, const path &__new_symlink)`
- void `std::filesystem::copy_symlink (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept`
- bool `std::filesystem::create_directories (const path &__p)`
- bool `std::filesystem::create_directories (const path &__p, error_code &__ec)`
- bool `std::filesystem::create_directory (const path &__p)`
- bool `std::filesystem::create_directory (const path &__p, const path &__attributes)`
- bool `std::filesystem::create_directory (const path &__p, const path &__attributes, error_code &__ec) noexcept`
- bool `std::filesystem::create_directory (const path &__p, error_code &__ec) noexcept`
- void `std::filesystem::create_directory_symlink (const path &__to, const path &__new_symlink)`
- void `std::filesystem::create_directory_symlink (const path &__to, const path &__new_symlink, error_code &__ec) noexcept`
- void `std::filesystem::create_hard_link (const path &__to, const path &__new_hard_link)`
- void `std::filesystem::create_hard_link (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept`



- void **std::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::filesystem::current\_path** ()
- void **std::filesystem::current\_path** (const [path](#) &\_\_p)
- void **std::filesystem::current\_path** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::filesystem::current\_path** ([error\\_code](#) &\_\_ec)
- bool **std::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool **std::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::exists** (const [path](#) &\_\_p)
- bool **std::filesystem::exists** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::exists** ([file\\_status](#)) noexcept
- uintmax\_t **std::filesystem::file\_size** (const [path](#) &\_\_p)
- uintmax\_t **std::filesystem::file\_size** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- uintmax\_t **std::filesystem::hard\_link\_count** (const [path](#) &\_\_p)
- uintmax\_t **std::filesystem::hard\_link\_count** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_block\_file** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_block\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_block\_file** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_character\_file** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_character\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_character\_file** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_directory** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_directory** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_empty** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_empty** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_fifo** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_fifo** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_fifo** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_other** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_other** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_other** ([file\\_status](#)) noexcept
- bool **std::filesystem::is\_regular\_file** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_regular\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_regular\_file** ([file\\_status](#)) noexcept
- bool **std::filesystem::is\_socket** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_socket** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_socket** ([file\\_status](#) \_\_s) noexcept
- bool **std::filesystem::is\_symlink** (const [path](#) &\_\_p)
- bool **std::filesystem::is\_symlink** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::filesystem::is\_symlink** ([file\\_status](#)) noexcept
- [file\\_time\\_type](#) **std::filesystem::last\_write\_time** (const [path](#) &\_\_p)
- [file\\_time\\_type](#) **std::filesystem::last\_write\_time** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::last\_write\_time** (const [path](#) &\_\_p, [file\\_time\\_type](#) \_\_new\_time)
- void **std::filesystem::last\_write\_time** (const [path](#) &\_\_p, [file\\_time\\_type](#) \_\_new\_time, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::permissions** (const [path](#) &\_\_p, [perms](#), [perm\\_options](#), [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::permissions** (const [path](#) &\_\_p, [perms](#) \_\_prms, [error\\_code](#) &\_\_ec) noexcept
- void **std::filesystem::permissions** (const [path](#) &\_\_p, [perms](#) \_\_prms, [perm\\_options](#) \_\_opts=perm\_options::replace)

- `path std::filesystem::proximate` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path std::filesystem::proximate` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::filesystem::proximate` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path std::filesystem::read_symlink` (const `path` &\_\_p)
- `path std::filesystem::read_symlink` (const `path` &\_\_p, `error_code` &\_\_ec)
- `path std::filesystem::relative` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path std::filesystem::relative` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::filesystem::relative` (const `path` &\_\_p, `error_code` &\_\_ec)
- `bool std::filesystem::remove` (const `path` &, `error_code` &) noexcept
- `bool std::filesystem::remove` (const `path` &\_\_p)
- `uintmax_t std::filesystem::remove_all` (const `path` &)
- `uintmax_t std::filesystem::remove_all` (const `path` &, `error_code` &)
- `void std::filesystem::rename` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::filesystem::rename` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `void std::filesystem::resize_file` (const `path` &\_\_p, `uintmax_t` \_\_size)
- `void std::filesystem::resize_file` (const `path` &\_\_p, `uintmax_t` \_\_size, `error_code` &\_\_ec) noexcept
- `space_info std::filesystem::space` (const `path` &\_\_p)
- `space_info std::filesystem::space` (const `path` &\_\_p, `error_code` &\_\_ec) noexcept
- `file_status std::filesystem::status` (const `path` &)
- `file_status std::filesystem::status` (const `path` &, `error_code` &) noexcept
- `bool std::filesystem::status_known` (`file_status`) noexcept
- `file_status std::filesystem::symlink_status` (const `path` &)
- `file_status std::filesystem::symlink_status` (const `path` &, `error_code` &) noexcept
- `path std::filesystem::temp_directory_path` ()
- `path std::filesystem::temp_directory_path` (`error_code` &\_\_ec)
- `path std::filesystem::weakly_canonical` (const `path` &\_\_p)
- `path std::filesystem::weakly_canonical` (const `path` &\_\_p, `error_code` &\_\_ec)

### 7.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 7.63 experimental/bits/fs\_ops.h File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Functions

- `path std::experimental::filesystem::absolute` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base, `error_code` &\_\_ec)
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, const `path` &\_\_base=current\_path())
- `path std::experimental::filesystem::canonical` (const `path` &\_\_p, `error_code` &\_\_ec)
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to)
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options)
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `copy_options` \_\_options, `error_code` &\_\_ec) noexcept
- `void std::experimental::filesystem::copy` (const `path` &\_\_from, const `path` &\_\_to, `error_code` &\_\_ec) noexcept
- `bool std::experimental::filesystem::copy_file` (const `path` &\_\_from, const `path` &\_\_to)

- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option)
- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, copy\_options \_\_option, [error\\_code](#) &\_\_ec)
- bool **std::experimental::filesystem::copy\_file** (const [path](#) &\_\_from, const [path](#) &\_\_to, [error\\_code](#) &\_\_ec)
- void **std::experimental::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↵  
symlink)
- void **std::experimental::filesystem::copy\_symlink** (const [path](#) &\_\_existing\_symlink, const [path](#) &\_\_new\_↵  
symlink, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::create\_directories** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::create\_directories** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes)
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, const [path](#) &\_\_attributes, [error\\_code](#)  
&\_\_ec) noexcept
- bool **std::experimental::filesystem::create\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::experimental::filesystem::create\_directory\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink,  
[error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link)
- void **std::experimental::filesystem::create\_hard\_link** (const [path](#) &\_\_to, const [path](#) &\_\_new\_hard\_link,  
[error\\_code](#) &\_\_ec) noexcept
- void **std::experimental::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink)
- void **std::experimental::filesystem::create\_symlink** (const [path](#) &\_\_to, const [path](#) &\_\_new\_symlink,  
[error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::experimental::filesystem::current\_path** ()
- void **std::experimental::filesystem::current\_path** (const [path](#) &\_\_p)
- void **std::experimental::filesystem::current\_path** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- [path](#) **std::experimental::filesystem::current\_path** ([error\\_code](#) &\_\_ec)
- bool **std::experimental::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2)
- bool **std::experimental::filesystem::equivalent** (const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) &\_\_ec) noex-  
cept
- bool **std::experimental::filesystem::exists** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::exists** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::exists** (file\_status \_\_s) noexcept
- uintmax\_t **std::experimental::filesystem::file\_size** (const [path](#) &\_\_p)
- uintmax\_t **std::experimental::filesystem::file\_size** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- uintmax\_t **std::experimental::filesystem::hard\_link\_count** (const [path](#) &\_\_p)
- uintmax\_t **std::experimental::filesystem::hard\_link\_count** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::is\_block\_file** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::is\_block\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::is\_block\_file** (file\_status \_\_s) noexcept
- bool **std::experimental::filesystem::is\_character\_file** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::is\_character\_file** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::is\_character\_file** (file\_status \_\_s) noexcept
- bool **std::experimental::filesystem::is\_directory** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::is\_directory** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::is\_directory** (file\_status \_\_s) noexcept
- bool **std::experimental::filesystem::is\_empty** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::is\_empty** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept
- bool **std::experimental::filesystem::is\_fifo** (const [path](#) &\_\_p)
- bool **std::experimental::filesystem::is\_fifo** (const [path](#) &\_\_p, [error\\_code](#) &\_\_ec) noexcept

- `bool std::experimental::filesystem::is_fifo (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_other (const path &__p)`
- `bool std::experimental::filesystem::is_other (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_other (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_regular_file (const path &__p)`
- `bool std::experimental::filesystem::is_regular_file (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_regular_file (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_socket (const path &__p)`
- `bool std::experimental::filesystem::is_socket (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_socket (file_status __s) noexcept`
- `bool std::experimental::filesystem::is_symlink (const path &__p)`
- `bool std::experimental::filesystem::is_symlink (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::is_symlink (file_status __s) noexcept`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p)`
- `file_time_type std::experimental::filesystem::last_write_time (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::experimental::filesystem::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `void std::experimental::filesystem::permissions (const path &__p, perms __prms)`
- `void std::experimental::filesystem::permissions (const path &__p, perms __prms, error_code &__ec) noexcept`
- `path std::experimental::filesystem::read_symlink (const path &__p)`
- `path std::experimental::filesystem::read_symlink (const path &__p, error_code &__ec)`
- `bool std::experimental::filesystem::remove (const path &__p)`
- `bool std::experimental::filesystem::remove (const path &__p, error_code &__ec) noexcept`
- `uintmax_t std::experimental::filesystem::remove_all (const path &__p)`
- `uintmax_t std::experimental::filesystem::remove_all (const path &__p, error_code &__ec)`
- `void std::experimental::filesystem::rename (const path &__from, const path &__to)`
- `void std::experimental::filesystem::rename (const path &__from, const path &__to, error_code &__ec) noexcept`
- `void std::experimental::filesystem::resize_file (const path &__p, uintmax_t __size)`
- `void std::experimental::filesystem::resize_file (const path &__p, uintmax_t __size, error_code &__ec) noexcept`
- `space_info std::experimental::filesystem::space (const path &__p)`
- `space_info std::experimental::filesystem::space (const path &__p, error_code &__ec) noexcept`
- `file_status std::experimental::filesystem::status (const path &__p)`
- `file_status std::experimental::filesystem::status (const path &__p, error_code &__ec) noexcept`
- `bool std::experimental::filesystem::status_known (file_status __s) noexcept`
- `file_status std::experimental::filesystem::symlink_status (const path &__p)`
- `file_status std::experimental::filesystem::symlink_status (const path &__p, error_code &__ec) noexcept`
- `path std::experimental::filesystem::system_complete (const path &__p)`
- `path std::experimental::filesystem::system_complete (const path &__p, error_code &__ec)`
- `path std::experimental::filesystem::temp_directory_path ()`
- `path std::experimental::filesystem::temp_directory_path (error_code &__ec)`

### 7.63.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 7.64 bits/fs\_path.h File Reference

### Classes

- class [std::filesystem::filesystem\\_error](#)
- class [std::filesystem::path::iterator](#)
- class [std::filesystem::path](#)

### Namespaces

- namespace [std](#)
- namespace [std::filesystem](#)

### Functions

- `size_t std::filesystem::hash_value (const path &__p) noexcept`

#### 7.64.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 7.65 experimental/bits/fs\_path.h File Reference

### Classes

- class [std::experimental::filesystem::v1::basic\\_string\\_view<\\_CharT, \\_Traits>](#)
- class [std::experimental::filesystem::v1::filesystem\\_error](#)
- class [std::experimental::filesystem::v1::path::iterator](#)
- class [std::experimental::filesystem::v1::path](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Functions

- `bool std::experimental::filesystem::operator< (const path &__lhs, const path &__rhs) noexcept`
- `bool std::experimental::filesystem::operator== (const path &__lhs, const path &__rhs) noexcept`

#### 7.65.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 7.66 fstream.tcc File Reference

### Namespaces

- namespace [std](#)

#### 7.66.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

## 7.67 funcref\_impl.h File Reference

### Classes

- class [std::function\\_ref<\\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>](#)

### Namespaces

- namespace [std](#)

#### 7.67.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.68 funtexcept.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- void [std::\\_\\_throw\\_bad\\_cast](#) (void)
- void [std::\\_\\_throw\\_bad\\_exception](#) (void)
- void [std::\\_\\_throw\\_bad\\_function\\_call](#) ()
- void [std::\\_\\_throw\\_bad\\_typeid](#) (void)
- void [std::\\_\\_throw\\_future\\_error](#) (int)
- void [std::\\_\\_throw\\_ios\\_failure](#) (const char \*)
- void [std::\\_\\_throw\\_ios\\_failure](#) (const char \*, int)
- void [std::\\_\\_throw\\_range\\_error](#) (const char \*)
- void [std::\\_\\_throw\\_system\\_error](#) (int)

#### 7.68.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

## 7.69 functional\_hash.h File Reference

### Classes

- struct [std::\\_\\_is\\_fast\\_hash<\\_Hash>](#)
- struct [std::hash<\\_Tp>](#)
- struct [std::hash<\\_Tp\\*>](#)
- struct [std::hash<bool>](#)
- struct [std::hash<char>](#)
- struct [std::hash<char16\\_t>](#)
- struct [std::hash<char32\\_t>](#)
- struct [std::hash<double>](#)
- struct [std::hash<float>](#)
- struct [std::hash<int>](#)
- struct [std::hash<long>](#)
- struct [std::hash<long double>](#)

- struct [std::hash< long long >](#)
- struct [std::hash< short >](#)
- struct [std::hash< signed char >](#)
- struct [std::hash< unsigned char >](#)
- struct [std::hash< unsigned int >](#)
- struct [std::hash< unsigned long >](#)
- struct [std::hash< unsigned long long >](#)
- struct [std::hash< unsigned short >](#)
- struct [std::hash< wchar\\_t >](#)

## Namespaces

- namespace [std](#)

## Macros

- `#define \_Cxx\_hashtable\_define\_trivial\_hash(_Tp)`

## Variables

- `template<typename _Tp, typename = void>`  
`constexpr bool std::\_\_is\_hash\_enabled\_for`
- `template<typename _Tp>`  
`constexpr bool std::\_\_is\_hash\_enabled\_for< _Tp, __void_t< decltype(hash< _Tp >()(declval< _Tp >()))>`  
`>`

### 7.69.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.70 funcwrap.h File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_MOF\_CV`

### Functions

- `template<typename _Fn>`  
`std::function\_ref (_Fn *) -> function_ref< _Fn >`
- `template<auto __fn, typename _Fn>`  
`std::function\_ref (constant_wrapper< __fn, _Fn >) -> function_ref< remove\_pointer\_t< _Fn > >`
- `template<auto __fn, typename _Fn, typename _Tp, typename _SignaturePtr = decltype(__polyfunc::__deduce_funcref< _Fn, _Tp >())>`  
`std::function\_ref (constant_wrapper< __fn, _Fn >, _Tp &&) -> function_ref< remove\_pointer\_t< _SignaturePtr > >`

### 7.70.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.71 `gslice.h` File Reference

### Classes

- class [std::gslice](#)

### Namespaces

- namespace [std](#)

#### 7.71.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.72 `gslice_array.h` File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)

### Namespaces

- namespace [std](#)

#### 7.72.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.73 `hash_bytes.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `size_t std::Fnv_hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)
- `size_t std::Hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)

#### 7.73.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.74 `backward/hashtable.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum { `_S_num_primes` }



## Functions

- unsigned long **\_\_gnu\_cxx::\_\_stl\_next\_prime** (unsigned long \_\_n)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All>  
bool **\_\_gnu\_cxx::operator!=** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All>  
bool **\_\_gnu\_cxx::operator==** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Extract, class \_EqKey, class \_All>  
void **\_\_gnu\_cxx::swap** (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)

## Variables

- template<typename \_PrimeType>  
const \_PrimeType **\_\_gnu\_cxx::Hashtable\_prime\_list< \_PrimeType >::\_\_stl\_prime\_list** [\_S\_num\_primes]

### 7.74.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 7.75 bits/hashtable.h File Reference

### Namespaces

- namespace [std](#)

### 7.75.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 7.76 hashtable\_policy.h File Reference

### Namespaces

- namespace [std](#)

### 7.76.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 7.77 indirect.h File Reference

### 7.77.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.78 indirect\_array.h File Reference

### Classes

- class [std::indirect\\_array< \\_Tp >](#)

## Namespaces

- namespace [std](#)

### 7.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.79 intcmp.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_equal (_Tp __t, _Up __u) noexcept`
- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_greater (_Tp __t, _Up __u) noexcept`
- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_greater_equal (_Tp __t, _Up __u) noexcept`
- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_less (_Tp __t, _Up __u) noexcept`
- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_less_equal (_Tp __t, _Up __u) noexcept`
- `template<typename _Tp, typename _Up>  
constexpr bool std::cmp_not_equal (_Tp __t, _Up __u) noexcept`
- `template<typename _Res, typename _Tp>  
constexpr bool std::in_range (_Tp __t) noexcept`

### 7.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 7.80 invoke.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>  
constexpr _Up && std::__invfwd (typename remove\_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>  
constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (_Callable &&__fn, _Args &&... __args)  
noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args) ← args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>  
constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`

- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > std::__invoke_r (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)`

### 7.80.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.81 ios\_base.h File Reference

### Classes

- class `std::ios_base::failure`
- class `std::ios_base`

### Namespaces

- namespace `std`

### Macros

- `#define _NOREPLACE_UNUSED`

### Enumerations

- enum `_ios_fmtflags` {  
`_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
`_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
`_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
`_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
`_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end`, `_S_ios_fmtflags_max`,  
`_S_ios_fmtflags_min` }
- enum `_ios_iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end`, `_S_ios_iostate_max`, `_S_ios_iostate_min` }
- enum `_ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_noreplace`, `_S_ios_openmode_end`,  
`_S_ios_openmode_max`, `_S_ios_openmode_min` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum class `std::io_errc` { `stream` }

### Functions

- `ios_base & std::boolalpha (ios_base &__base)`
- `ios_base & std::dec (ios_base &__base)`
- `ios_base & std::defaultfloat (ios_base &__base)`
- `ios_base & std::fixed (ios_base &__base)`

- `ios_base & std::hex (ios_base & __base)`
- `ios_base & std::hexfloat (ios_base & __base)`
- `ios_base & std::internal (ios_base & __base)`
- `const error_category & std::iostream_category () noexcept`
- `ios_base & std::left (ios_base & __base)`
- `error_code std::make_error_code (io_errc __e) noexcept`
- `error_condition std::make_error_condition (io_errc __e) noexcept`
- `ios_base & std::noboolalpha (ios_base & __base)`
- `ios_base & std::noshowbase (ios_base & __base)`
- `ios_base & std::noshowpoint (ios_base & __base)`
- `ios_base & std::noshowpos (ios_base & __base)`
- `ios_base & std::noskipws (ios_base & __base)`
- `ios_base & std::nounitbuf (ios_base & __base)`
- `ios_base & std::nouppercase (ios_base & __base)`
- `ios_base & std::oct (ios_base & __base)`
- `constexpr _ios_Fmtflags std::operator& (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_losestate std::operator& (_ios_losestate __a, _ios_losestate __b) noexcept`
- `constexpr _ios_Openmode std::operator& (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator&= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_losestate & std::operator&= (_ios_losestate & __a, _ios_losestate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator&= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_losestate std::operator^ (_ios_losestate __a, _ios_losestate __b) noexcept`
- `constexpr _ios_Openmode std::operator^ (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator^= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_losestate & std::operator^= (_ios_losestate & __a, _ios_losestate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator^= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator| (_ios_Fmtflags __a, _ios_Fmtflags __b) noexcept`
- `constexpr _ios_losestate std::operator| (_ios_losestate __a, _ios_losestate __b) noexcept`
- `constexpr _ios_Openmode std::operator| (_ios_Openmode __a, _ios_Openmode __b) noexcept`
- `constexpr const _ios_Fmtflags & std::operator|= (_ios_Fmtflags & __a, _ios_Fmtflags __b) noexcept`
- `constexpr const _ios_losestate & std::operator|= (_ios_losestate & __a, _ios_losestate __b) noexcept`
- `constexpr const _ios_Openmode & std::operator|= (_ios_Openmode & __a, _ios_Openmode __b) noexcept`
- `constexpr _ios_Fmtflags std::operator~ (_ios_Fmtflags __a) noexcept`
- `constexpr _ios_losestate std::operator~ (_ios_losestate __a) noexcept`
- `constexpr _ios_Openmode std::operator~ (_ios_Openmode __a) noexcept`
- `ios_base & std::right (ios_base & __base)`
- `ios_base & std::scientific (ios_base & __base)`
- `ios_base & std::showbase (ios_base & __base)`
- `ios_base & std::showpoint (ios_base & __base)`
- `ios_base & std::showpos (ios_base & __base)`
- `ios_base & std::skipws (ios_base & __base)`
- `ios_base & std::unitbuf (ios_base & __base)`
- `ios_base & std::uppercase (ios_base & __base)`

### 7.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 7.82 iosfwd\_file.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef [basic\\_filebuf](#)< char > [std::filebuf](#)
- typedef [basic\\_fstream](#)< char > [std::fstream](#)
- typedef [basic\\_ifstream](#)< char > [std::ifstream](#)
- typedef [basic\\_ofstream](#)< char > [std::ofstream](#)
- typedef [basic\\_filebuf](#)< wchar\_t > [std::wfilebuf](#)
- typedef [basic\\_fstream](#)< wchar\_t > [std::wfstream](#)
- typedef [basic\\_ifstream](#)< wchar\_t > [std::wifstream](#)
- typedef [basic\\_ofstream](#)< wchar\_t > [std::wofstream](#)

#### 7.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.83 iosfwd\_span.h File Reference

### 7.83.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.84 iosfwd\_string.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef [basic\\_istringstream](#)< char > [std::istringstream](#)
- typedef [basic\\_ostringstream](#)< char > [std::ostringstream](#)
- typedef [basic\\_stringbuf](#)< char > [std::stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [std::stringstream](#)
- typedef [basic\\_istringstream](#)< wchar\_t > [std::wistringstream](#)
- typedef [basic\\_ostringstream](#)< wchar\_t > [std::wostringstream](#)
- typedef [basic\\_stringbuf](#)< wchar\_t > [std::wstringbuf](#)
- typedef [basic\\_stringstream](#)< wchar\_t > [std::wstringstream](#)

#### 7.84.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.85 iosfwd\_sync.h File Reference

### 7.85.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.86 istream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits>  
void std::__istream_extract (basic\_istream< _CharT, _Traits > &, _CharT *, streamsize)`
- `template<typename _CharT, typename _Traits>  
basic\_istream< _CharT, _Traits > & std::ws (basic\_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits>  
basic\_istream< _CharT, _Traits > & std::operator>> (basic\_istream< _CharT, _Traits > &__in, _CharT &__c)`

### 7.86.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

## 7.87 iterator\_concepts.h File Reference

### Classes

- struct [std::default\\_sentinel\\_t](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Concepts

- concept [std::indirectly\\_comparable](#)
- concept [std::indirectly\\_copyable](#)
- concept [std::indirectly\\_movable](#)
- concept [std::indirectly\\_readable](#)
- concept [std::indirectly\\_swappable](#)
- concept [std::indirectly\\_writable](#)
- concept [std::mergeable](#)
- concept [std::permutable](#)
- concept [std::sortable](#)
- concept [std::weakly\\_incrementable](#)

### Typedefs

- `template<typename _Tp>  
using std::__indirect_value_t`
- `template<typename _Iter>  
using std::__detail::__iter_concept`
- `template<typename _Tp>  
using std::__detail::__iter_diff_t`
- `template<typename _Iter, typename _Tp = _Iter>  
using std::__detail::__iter_traits`

- `template<typename _Tp>`  
  using `std::__detail::__iter_value_t`
- `template<typename _Tp>`  
  using `std::__detail::__range_iter_t`
- `template<typename _Tp>`  
  using `std::__detail::__with_ref`
- `template<typename _Fn, typename... _Is>`  
  using `std::indirect_result_t`
- `template<indirectly_readable _Tp>`  
  using `std::iter_common_reference_t`
- `template<typename _Tp>`  
  using `std::iter_difference_t`
- `template<__detail::__dereferenceable _Tp>`  
  using `std::iter_reference_t`
- `template<__detail::__dereferenceable _Tp>`  
  using `std::iter_rvalue_reference_t`
- `template<typename _Tp>`  
  using `std::iter_value_t`
- `template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>`  
  using `std::projected`
- `template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>`  
  using `std::projected_value_t`

## Functions

- `template<typename _Tp>`  
  constexpr auto `std::ranges::__access::__begin` (`_Tp &__t`)
- `void std::ranges::__access::begin` ()=delete

## Variables

- constexpr `default_sentinel_t std::default_sentinel`
- `template<typename _Sent, typename _Iter>`  
  constexpr bool `std::disable_sized_sentinel_for`
- constexpr `__imove::__IterMove std::ranges::iter_move`
- constexpr `__iswap::__IterSwap std::ranges::iter_swap`
- constexpr `unreachable_sentinel_t std::unreachable_sentinel`

### 7.87.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.88 list.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX20_ONLY(__expr)`

### 7.88.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 7.89 locale\_classes.h File Reference

### Classes

- class `std::__cxx11::collate<_CharT>`
- class `std::__cxx11::collate_byname<_CharT>`
- class `std::locale::facet`
- class `std::locale::id`
- class `std::locale`

### Namespaces

- namespace `std`

### Variables

- template<typename `_CharT`>  
`locale::id std::collate<_CharT>::id`

### 7.89.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.90 locale\_classes.tcc File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_STD_FACET(...)`

### Functions

- template<typename `_Facet`>  
`const _Facet * std::__try_use_facet` (const `locale` &\_\_loc)
- template<typename `_Facet`>  
`bool std::has_facet` (const `locale` &\_\_loc)
- template<typename `_Facet`>  
`const _Facet & std::use_facet` (const `locale` &\_\_loc)

### 7.90.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.



## 7.91 locale\_conv.h File Reference

### Classes

- class [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>](#)
- class [std::wstring\\_convert<\\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc>](#)

### Namespaces

- namespace [std](#)

#### 7.91.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.92 locale\_facets.h File Reference

### Classes

- class [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar\\_t>](#)
- class [std::ctype\\_byname<\\_CharT>](#)
- class [std::ctype\\_byname<char>](#)
- class [std::num\\_get<\\_CharT, \\_Inlter>](#)
- class [std::num\\_put<\\_CharT, \\_Outlter>](#)
- class [std::numpunct<\\_CharT>](#)
- class [std::numpunct\\_byname<\\_CharT>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_NUM_CXX11_FACETS`
- `#define _GLIBCXX_NUM_FACETS`
- `#define _GLIBCXX_NUM_LBDL_ALT128_FACETS`
- `#define _GLIBCXX_NUM_UNICODE_FACETS`

### Functions

- `template<typename _CharT>`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp>`  
`void std::__convert_to_v (const char *, _Tp &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, float &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<> void std::__convert_to_v (const char *, long double &, ios\_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT, typename _Outlter>`  
`_Outlter std::__write (_Outlter __s, const _CharT *__ws, int __len)`

- `template<typename _CharT>`  
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _CharT>`  
`bool std::isalnum ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isalpha ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isblank ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::iscntrl ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isdigit ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isgraph ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::islower ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isprint ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::ispunct ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isspace ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isupper ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isxdigit ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::tolower ( _CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::toupper ( _CharT __c, const locale &__loc)`

## Variables

- `template<typename _CharT>`  
`locale::id std::ctype< _CharT >::id`
- `template<typename _CharT, typename _Inlter>`  
`locale::id std::num_get< _CharT, _Inlter >::id`
- `template<typename _CharT, typename _Outlter>`  
`locale::id std::num_put< _CharT, _Outlter >::id`
- `template<typename _CharT>`  
`locale::id std::numpunct< _CharT >::id`

### 7.92.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.93 locale\_facets.tcc File Reference

### Namespaces

- namespace `std`

## Functions

- `template<typename _CharT>`  
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT * __first, const _CharT * __last)`
- `template<typename _CharT, typename _ValueT>`  
`int std::__int_to_char (_CharT * __bufend, _ValueT __v, const _CharT * __lit, ios\_base::fmtflags __flags, bool __dec)`
- `bool std::__verify_grouping (const char * __grouping, size_t __grouping_size, const string & __grouping_tmp) throw ()`

### 7.93.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.94 locale\_facets\_nonio.h File Reference

### Classes

- class [std::messages< \\_CharT >](#)
- struct [std::messages\\_base](#)
- class [std::messages\\_byname< \\_CharT >](#)
- class [std::money\\_base](#)
- class [std::money\\_get< \\_CharT, \\_InIter >](#)
- class [std::money\\_put< \\_CharT, \\_OutIter >](#)
- class [std::moneypunct< \\_CharT, \\_Intl >](#)
- class [std::moneypunct\\_byname< \\_CharT, \\_Intl >](#)
- class [std::time\\_base](#)
- class [std::time\\_get< \\_CharT, \\_InIter >](#)
- class [std::time\\_get\\_byname< \\_CharT, \\_InIter >](#)
- class [std::time\\_put< \\_CharT, \\_OutIter >](#)
- class [std::time\\_put\\_byname< \\_CharT, \\_OutIter >](#)

### Namespaces

- namespace [std](#)

### Variables

- `template<typename _CharT>`  
[locale::id](#) **std::\_\_timepunct< \_CharT >::id**
- `template<typename _CharT>`  
`const _CharT * std::__timepunct_cache< _CharT >::S_timezones [14]`
- `template<> const char * std::__timepunct_cache< char >::S_timezones [14]`
- `template<> const wchar_t * std::__timepunct_cache< wchar_t >::S_timezones [14]`
- `template<typename _CharT>`  
[locale::id](#) **std::messages< \_CharT >::id**
- `template<typename _CharT, typename _InIter>`  
[locale::id](#) **std::money\_get< \_CharT, \_InIter >::id**
- `template<typename _CharT, typename _OutIter>`  
[locale::id](#) **std::money\_put< \_CharT, \_OutIter >::id**
- `template<typename _CharT, bool _Intl>`  
[locale::id](#) **std::moneypunct< \_CharT, \_Intl >::id**

- `template<typename _CharT, bool _Intl>`  
`const bool std::moneypunct< _CharT, _Intl >::intl`
- `template<typename _CharT, bool _Intl>`  
`const bool std::moneypunct_byname< _CharT, _Intl >::intl`
- `template<typename _CharT, typename _Inlter>`  
`locale::id std::time_get< _CharT, _Inlter >::id`
- `template<typename _CharT, typename _Outlter>`  
`locale::id std::time_put< _CharT, _Outlter >::id`

#### 7.94.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 7.95 `locale_facets_nonio.tcc` File Reference

#### Namespaces

- namespace [std](#)

#### 7.95.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 7.96 `localefwd.h` File Reference

#### Namespaces

- namespace [std](#)

#### Functions

- `template<typename _Facet>`  
`bool std::has_facet (const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isspace (_CharT __c, const locale &__loc)`

- `template<typename _CharT>`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet>`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 7.96.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 7.97 `mask_array.h` File Reference

### Classes

- class [std::mask\\_array< \\_Tp >](#)

### Namespaces

- namespace [std](#)

### 7.97.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.98 `max_size_type.h` File Reference

### 7.98.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.99 `memory_resource.h` File Reference

### Classes

- struct [std::allocator\\_traits< pmr::polymorphic\\_allocator< \\_Tp > >](#)
- class [std::pmr::memory\\_resource](#)
- class [std::pmr::polymorphic\\_allocator< \\_Tp >](#)

### Namespaces

- namespace [std](#)

### Functions

- `bool std::pmr::operator== (const memory\_resource &__a, const memory\_resource &__b) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::pmr::operator== (const polymorphic\_allocator< _Tp1 > &__a, const polymorphic\_allocator< _Tp2 > &__b) noexcept`

### 7.99.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory_resource>`.

## 7.100 `memoryfwd.h` File Reference

### Namespaces

- namespace [std](#)

### 7.100.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.101 `mofunc_impl.h` File Reference

### Classes

- class [std::move\\_only\\_function<\\_Res\(\\_ArgTypes...\) \\_GLIBCXX\\_MOF\\_CV noexcept\(\\_Noex\)>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_MOF_CV_REF`
- `#define _GLIBCXX_MOF_INV_QUALS`
- `#define _GLIBCXX_MOF_REF`

### 7.101.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.102 `monostate.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- constexpr strong\_ordering **std::operator<=>** (monostate, monostate) noexcept
- constexpr bool **std::operator==** (monostate, monostate) noexcept

### 7.102.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 7.103 `move.h` File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_FWDREF(_Tp)`
- `#define _GLIBCXX_MOVE(__val)`

## Typedefs

- `template<typename _Tp, typename _Up>`  
`using std::\_\_like\_t`

## Functions

- `template<typename _Tp>`  
`constexpr _Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp std::\_\_exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _Tp>`  
`constexpr _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr __like_t< _Tp, _Up > std::forward\_like (_Up &&__x) noexcept`
- `template<typename _Tp>`  
`constexpr std::remove\_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp>`  
`constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move\_if\_noexcept (_Tp &__x) noexcept`
- `template<typename _Tp>`  
`constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/\*conditional \*/) is\_nothrow\_move\_assignable< _Tp > >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/\*conditional \*/)`

### 7.103.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 7.104 nested\_exception.h File Reference

### Classes

- class [std::nested\\_exception](#)

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Ex>`  
`void std::rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Tp>`  
`void std::throw\_with\_nested (_Tp &&__t)`

### 7.104.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 7.105 `bits/new_allocator.h` File Reference

### Classes

- class [std::\\_\\_new\\_allocator<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_OPERATOR_DELETE`
- `#define _GLIBCXX_OPERATOR_NEW`
- `#define _GLIBCXX_SIZED_DEALLOC(p, n)`

### 7.105.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.106 `ext/new_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::new\\_allocator<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 7.106.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.107 `new_except.h` File Reference

### Classes

- class [std::bad\\_alloc](#)

### Namespaces

- namespace [std](#)



### 7.107.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<new>`.

## 7.108 new\_throw.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void)
- void **std::\_\_throw\_bad\_array\_new\_length** (void)

### 7.108.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<new>`.

## 7.109 node\_handle.h File Reference

### 7.109.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map,set,unordered_map,unordered_set>`.

## 7.110 ostream.h File Reference

### Classes

- class [std::basic\\_ostream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Os, typename _Tp>`  
using **std::\_\_rvalue\_stream\_insertion\_t**

### Functions

- `template<typename _Ostream, typename _Tp>`  
`__rvalue_stream_insertion_t<_Ostream, _Tp> std::operator<< (_Ostream &&__os, const _Tp &__x)`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & std::operator<< (basic\_ostream< char, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & std::operator<< (basic\_ostream< char, _Traits > &, char32_t)=delete`
- `template<typename _Traits>`  
`basic\_ostream< char, _Traits > & std::operator<< (basic\_ostream< char, _Traits > &, const char16_t *)=delete`

- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const char32_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete`
- `template<typename _Traits>`  
`basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete`
  
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, signed char __c)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, const char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, const signed char *__s)`
- `template<typename _Traits>`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > & __out, const unsigned char *__s)`

### 7.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 7.111 ostream.tcc File Reference

### Namespaces

- namespace `std`

## Functions

- `template<typename _CharT, typename _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char`  
`*__s)`

### 7.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 7.112 ostream\_insert.h File Reference

### Namespaces

- namespace `std`

### 7.112.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 7.113 ostream\_print.h File Reference

### 7.113.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

This file contains the parts of `<ostream>` which are currently defined inline, but should be moved into the library eventually.

## 7.114 out\_ptr.h File Reference

### Classes

- class `std::inout_ptr_t< _Smart, _Pointer, _Args >`
- class `std::out_ptr_t< _Smart, _Pointer, _Args >`

### Namespaces

- namespace `std`

## Functions

- `template<typename _Pointer = void, typename _Smart, typename... _Args>`  
`auto std::inout_ptr (_Smart &__s, _Args &&... __args)`
- `template<typename _Pointer = void, typename _Smart, typename... _Args>`  
`auto std::out_ptr (_Smart &__s, _Args &&... __args)`

### 7.114.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.115 `parse_numbers.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<unsigned long long _Val>`  
using [std::\\_\\_parse\\_int::\\_\\_ull\\_constant](#)
- `template<char... _Digs>`  
using [std::\\_\\_select\\_int::\\_\\_Select\\_int](#)

#### 7.115.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

## 7.116 `postypes.h` File Reference

### Classes

- class [std::fpos<\\_StateT>](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `typedef long long` [std::streamoff](#)
- `typedef` [fpos<mbstate\\_t>](#) [std::streampos](#)
- `typedef ptrdiff_t` [std::streamsize](#)
- `typedef` [fpos<mbstate\\_t>](#) [std::u16streampos](#)
- `typedef` [fpos<mbstate\\_t>](#) [std::u32streampos](#)
- `typedef` [fpos<mbstate\\_t>](#) [std::wstreampos](#)

### Functions

- `template<typename _StateT>`  
bool [std::operator!=](#) (const [fpos<\\_StateT>](#) &\_\_lhs, const [fpos<\\_StateT>](#) &\_\_rhs)
- `template<typename _StateT>`  
bool [std::operator==](#) (const [fpos<\\_StateT>](#) &\_\_lhs, const [fpos<\\_StateT>](#) &\_\_rhs)

#### 7.116.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.117 `predefined_ops.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Typedefs

- `template<typename _Fn>`  
using `__gnu_cxx::__ops::__by_ref_or_value_fn`
- `template<typename _Func, typename _Value>`  
using `__gnu_cxx::__ops::__Comp_with_val_1st`
- `template<typename _Func, typename _Value>`  
using `__gnu_cxx::__ops::__Comp_with_val_2nd`
- `typedef std::equal_to< void > __gnu_cxx::__ops::equal_to`
- `typedef std::less< void > __gnu_cxx::__ops::less`

## Functions

- `template<typename _Value>`  
`constexpr __Comp_with_val_2nd< equal_to, _Value > __gnu_cxx::__ops::__equal_to (const _Value &__val)`
- `template<typename _Func, typename _Value>`  
`constexpr __Comp_with_val_1st< _Func, _Value > __gnu_cxx::__ops::bind1st (_Func &__f, const _Value &__val)`
- `template<typename _Func, typename _Value>`  
`constexpr __Comp_with_val_2nd< _Func, _Value > __gnu_cxx::__ops::bind2nd (_Func &__f, const _Value &__val)`
- `template<typename _Func>`  
`constexpr __Unary_negate< _Func > __gnu_cxx::__ops::not1 (_Func &__f)`

### 7.117.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include `<algorithm>`.

## 7.118 `print.h` File Reference

### 7.118.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<print>`.

This file contains the parts of `<print>` which are currently defined inline, but should be moved into the library eventually.

## 7.119 `ptr_traits.h` File Reference

### Classes

- struct `std::pointer_traits< _Ptr >`
- struct `std::pointer_traits< _Tp * >`

### Namespaces

- namespace `__gnu_debug`
- namespace `std`

### Typedefs

- `template<typename _Ptr, typename _Tp>`  
using `std::__ptr_rebind`

## Functions

- `template<typename _Tp>`  
`constexpr _Tp * std::to\_address (_Tp *__ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr auto std::to\_address (const _Ptr &__ptr) noexcept`

### 7.119.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.120 `quoted_string.h` File Reference

### Classes

- struct [std::\\_\\_detail::Quoted\\_string](#)< \_String, \_CharT >

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _String>`  
`std::basic\_ostream< _CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const Quoted\_string< _String, _CharT > &__str)`
- `template<typename _CharT, typename _Traits>`  
`std::basic\_ostream< _CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const Quoted\_string< const _CharT *, _CharT > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic\_istream< _CharT, _Traits > & std::\_\_detail::operator>> (std::basic\_istream< _CharT, _Traits > &__is, const Quoted\_string< basic\_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)`

### 7.120.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iomanip>`.

## 7.121 `random.h` File Reference

### Classes

- class [std::bernoulli\\_distribution](#)
- class [std::binomial\\_distribution](#)< \_IntType >
- class [std::cauchy\\_distribution](#)< \_RealType >
- class [std::chi\\_squared\\_distribution](#)< \_RealType >
- class [std::discard\\_block\\_engine](#)< \_RandomNumberEngine, \_\_p, \_\_r >
- class [std::discrete\\_distribution](#)< \_IntType >
- class [std::exponential\\_distribution](#)< \_RealType >
- class [std::extreme\\_value\\_distribution](#)< \_RealType >
- class [std::fisher\\_f\\_distribution](#)< \_RealType >
- class [std::gamma\\_distribution](#)< \_RealType >
- class [std::geometric\\_distribution](#)< \_IntType >

- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m >`
- class `std::lognormal_distribution<_RealType >`
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::negative_binomial_distribution<_IntType >`
- class `std::normal_distribution<_RealType >`
- struct `std::uniform_real_distribution<_RealType >::param_type`
- struct `std::normal_distribution<_RealType >::param_type`
- struct `std::lognormal_distribution<_RealType >::param_type`
- struct `std::gamma_distribution<_RealType >::param_type`
- struct `std::chi_squared_distribution<_RealType >::param_type`
- struct `std::cauchy_distribution<_RealType >::param_type`
- struct `std::fisher_f_distribution<_RealType >::param_type`
- struct `std::student_t_distribution<_RealType >::param_type`
- struct `std::bernoulli_distribution::param_type`
- struct `std::binomial_distribution<_IntType >::param_type`
- struct `std::geometric_distribution<_IntType >::param_type`
- struct `std::negative_binomial_distribution<_IntType >::param_type`
- struct `std::poisson_distribution<_IntType >::param_type`
- struct `std::exponential_distribution<_RealType >::param_type`
- struct `std::weibull_distribution<_RealType >::param_type`
- struct `std::extreme_value_distribution<_RealType >::param_type`
- struct `std::discrete_distribution<_IntType >::param_type`
- struct `std::piecewise_constant_distribution<_RealType >::param_type`
- struct `std::piecewise_linear_distribution<_RealType >::param_type`
- class `std::philox_engine<_UIntType, __w, __n, __r, __consts >`
- class `std::piecewise_constant_distribution<_RealType >`
- class `std::piecewise_linear_distribution<_RealType >`
- class `std::poisson_distribution<_IntType >`
- class `std::random_device`
- class `std::seed_seq`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k >`
- class `std::student_t_distribution<_RealType >`
- class `std::subtract_with_carry_engine<_UIntType, __w, __s, __r >`
- class `std::uniform_real_distribution<_RealType >`
- class `std::weibull_distribution<_RealType >`

## Namespaces

- namespace `std`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` `std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` `std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` `std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL >` `std::mt19937_64`

- typedef `philox_engine`< uint\_fast32\_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > `std::philox4x32`
- typedef `philox_engine`< uint\_fast64\_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470←EE14C6C93, 0xBB67AE8584CAA73B > `std::philox4x64`
- typedef `discard_block_engine`< ranlux24\_base, 223, 23 > `std::ranlux24`
- typedef `subtract_with_carry_engine`< uint\_fast32\_t, 24, 10, 24 > `std::ranlux24_base`
- typedef `discard_block_engine`< ranlux48\_base, 389, 11 > `std::ranlux48`
- typedef `subtract_with_carry_engine`< uint\_fast64\_t, 48, 5, 12 > `std::ranlux48_base`

## Functions

- template<typename \_RealType, size\_t \_\_bits, typename \_UniformRandomNumberGenerator>  
\_RealType `std::generate_canonical` (\_UniformRandomNumberGenerator &\_\_g)
- template<typename \_IntType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &, const `std::uniform_int_distribution`< \_IntType > &)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &, const `std::uniform_real_distribution`< \_RealType > &)
- template<typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::bernoulli_distribution` &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::cauchy_distribution`< \_RealType > &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::exponential_distribution`< \_RealType > &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::extreme_value_distribution`< \_RealType > &\_\_x)
- template<typename \_IntType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::geometric_distribution`< \_IntType > &\_\_x)
- template<typename \_RandomNumberEngine, size\_t \_\_w, typename \_UIntType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::independent_bits_engine`< \_RandomNumberEngine, \_\_w, \_UIntType > &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_ostream`< \_CharT, \_Traits > & `std::operator<<` (`std::basic_ostream`< \_CharT, \_Traits > &\_\_os, const `std::weibull_distribution`< \_RealType > &\_\_x)
- template<typename \_IntType, typename \_CharT, typename \_Traits>  
`std::basic_istream`< \_CharT, \_Traits > & `std::operator>>` (`std::basic_istream`< \_CharT, \_Traits > &, `std::uniform_int_distribution`< \_IntType > &)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_istream`< \_CharT, \_Traits > & `std::operator>>` (`std::basic_istream`< \_CharT, \_Traits > &, `std::uniform_real_distribution`< \_RealType > &)
- template<typename \_CharT, typename \_Traits>  
`std::basic_istream`< \_CharT, \_Traits > & `std::operator>>` (`std::basic_istream`< \_CharT, \_Traits > &\_\_is, `std::bernoulli_distribution` &\_\_x)
- template<typename \_RealType, typename \_CharT, typename \_Traits>  
`std::basic_istream`< \_CharT, \_Traits > & `std::operator>>` (`std::basic_istream`< \_CharT, \_Traits > &\_\_is, `std::cauchy_distribution`< \_RealType > &\_\_x)



- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::weibull_distribution< _RealType > &__x)`

### 7.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 7.122 bits/random.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_GEN_CANON_CONST`

### Functions

- `template<typename _ValT, typename _CharT, typename _Traits>  
basic_istream< _CharT, _Traits > & std::__detail::__extract_params (basic_istream< _CharT, _Traits > &__is,  
vector< _ValT > &__vals, size_t __n)`
- `template<typename _UInt1, typename _UInt2, typename _UComm = __conditional_t<(sizeof(_UInt2) > sizeof(_UInt1)), _UInt2, _UInt1>>  
constexpr __gen_canon_log_res< _UInt1 > std::__gen_canon_log (_UInt1 __val, _UInt2 __base)`
- `template<typename _RealT, size_t __d, typename _Urbg>  
_RealT std::__generate_canonical_any (_Urbg &__urng)`
- `template<typename _RealT, size_t __d, typename _Urbg>  
_RealT std::__generate_canonical_pow2 (_Urbg &__urng)`
- `template<typename _Rng>  
constexpr bool std::__is_power_of_2_less_1 (_Rng __range)`
- `template<typename _InputIterator, typename _RealType>  
bool std::__detail::__load_first2 (_InputIterator &__first, _InputIterator __last, _RealType *__out)`
- `template<typename _Tp>  
constexpr bool std::__detail::__p1_representable_as_double (_Tp __x) noexcept`
- `template<typename _Tp>  
constexpr bool std::__detail::__representable_as_double (_Tp __x) noexcept`
- `template<typename _RealT, size_t __digits, typename _Urbg>  
_RealT std::generate_canonical (_Urbg &__urng)`
- `template<typename _IntType, typename _CharT, typename _Traits>  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const  
std::uniform_int_distribution< _IntType > &)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const binomial_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const chi_squared_distribution< _RealType > & __x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const discard_block_engine< _RandomNumberEngine, __p, __r > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const discrete_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const fisher_f_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const gamma_distribution< _RealType > & __x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const linear_congruential_engine< _UIntType, __a, __c, __m > & __lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const lognormal_distribution< _RealType > & __x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, ↵`  
`_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f`  
`> & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const negative_binomial_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const piecewise_constant_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const piecewise_linear_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const poisson_distribution< _IntType > & __x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os,`  
`const shuffle_order_engine< _RandomNumberEngine, __k > & __x)`
- `template<typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const`  
`std::bernoulli_distribution & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const student_t_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RealType>`  
`bool std::operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`chi_squared_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`lognormal_distribution< _RealType > &__x)`

- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`& __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`negative_binomial_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`piecewise_constant_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`piecewise_linear_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`poisson_distribution< _IntType > & __x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`shuffle_order_engine< _RandomNumberEngine, __k > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`std::cauchy_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`std::exponential_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`std::extreme_value_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`std::geometric_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`std::weibull_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`student_t_distribution< _RealType > & __x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is,`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > & __x)`

## Variables

- `template<typename _Tp>`  
`const bool std::__is_rand_dist_float_v`

### 7.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 7.123 ext/random.tcc File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::beta\_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::hypergeometric\_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::normal\_mv\_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t  
__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT,  
typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
&__os, const \_\_gnu\_cxx::simd\_fast\_mersenne\_twister\_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1,  
__sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::triangular\_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::uniform\_inside\_sphere\_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::uniform\_on\_sphere\_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const \_\_gnu\_cxx::von\_mises\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const arcsine\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const hoyt\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const k\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const logistic\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const nakagami\_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>  
std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_cxx::operator<< (std::basic\_ostream< _CharT, _Traits > &↵  
__os, const pareto\_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &↵`  
`__os, const rice_distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType>`  
`bool __gnu_cxx::operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const`  
`__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool __gnu_cxx::operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __↵`  
`pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4`  
`> &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __↵`  
`sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<typename _UIntType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT,`  
`typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2,`  
`__msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, logistic_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, nakagami_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, rice_distribution< _RealType > &__x)`

### 7.123.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/random>`.

## 7.124 range\_access.h File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _Container>`  
`constexpr auto std::begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::begin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.↵`  
`begin())`
- `template<typename _Container>`  
`constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(↵`  
`__cont))`
- `template<typename _Container>`  
`constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(↵`  
`__cont))`
- `template<typename _Container>`  
`constexpr auto std::crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↵`  
`__cont))`
- `template<typename _Container>`  
`constexpr auto std::crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↵`  
`__cont))`
- `template<typename _Container>`  
`constexpr auto std::data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::data (_Tp(&__array)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.↵`  
`data())`
- `template<typename _Container>`  
`constexpr auto std::empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__↵`  
`cont.empty())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool std::empty (const _Tp(&)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`



- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::end (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())`
- `template<typename _Container>`  
`constexpr auto std::rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.↵  
rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse\_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__↵  
cont.rbegin())`
- `template<typename _Tp>`  
`constexpr reverse\_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse\_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.↵  
rend())`
- `template<typename _Tp>`  
`constexpr reverse\_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il) noexcept`
- `template<typename _Container>`  
`constexpr auto std::size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr size_t std::size (const _Tp(&)[_Nm]) noexcept`
- `template<typename _Container>`  
`constexpr auto std::ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common\_type\_t<  
ptrdiff_t, make\_signed\_t< decltype(__cont.size())> >`
- `template<typename _Tp, ptrdiff_t _Num>`  
`constexpr ptrdiff_t std::ssize (const _Tp(&)[_Num]) noexcept`

### 7.124.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.125 `ranges_algo.h` File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Fp>`  
`using std::ranges::\_\_detail::\_\_by\_ref\_or\_value\_fn`
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
`using std::ranges::binary\_transform\_result`
- `template<typename _Iter, typename _Out>`  
`using std::ranges::copy\_if\_result`
- `template<typename _Iter, typename _Tp>`  
`using std::ranges::fold\_left\_first\_with\_iter\_result`



- `template<typename _Iter, typename _Tp>`  
  using **`std::ranges::fold_left_with_iter_result`**
- `template<typename _Iter, typename _Fp>`  
  using **`std::ranges::for_each_n_result`**
- `template<typename _Iter, typename _Fp>`  
  using **`std::ranges::for_each_result`**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
  using **`std::ranges::merge_result`**
- `template<typename _Iter>`  
  using **`std::ranges::minmax_element_result`**
- `template<typename _Tp>`  
  using **`std::ranges::minmax_result`**
- `template<typename _Iter>`  
  using **`std::ranges::next_permutation_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::partial_sort_copy_result`**
- `template<typename _Iter, typename _Out1, typename _Out2>`  
  using **`std::ranges::partition_copy_result`**
- `template<typename _Iter>`  
  using **`std::ranges::prev_permutation_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::remove_copy_if_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::remove_copy_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::replace_copy_if_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::replace_copy_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::reverse_copy_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::rotate_copy_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::set_difference_result`**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
  using **`std::ranges::set_intersection_result`**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
  using **`std::ranges::set_symmetric_difference_result`**
- `template<typename _Iter1, typename _Iter2, typename _Out>`  
  using **`std::ranges::set_union_result`**
- `template<typename _Iter1, typename _Iter2>`  
  using **`std::ranges::swap_ranges_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::unary_transform_result`**
- `template<typename _Iter, typename _Out>`  
  using **`std::ranges::unique_copy_result`**

## Functions

- `template<typename _Iter, typename _Comp>`  
  constexpr void **`std::ranges::detail::__adjust_heap`** (`_Iter __first`, `iter_difference_t<_Iter> __holeIndex`,  
  `iter_difference_t<_Iter> __len`, `iter_value_t<_Iter> __value`, `_Comp __comp`)

- `template<typename _Iter, typename _Distance, typename _Compare>`  
`constexpr void std::ranges::__detail::__chunk_insertion_sort (_Iter __first, _Iter __last, _Distance __chunk←`  
`__size, _Compare __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__final_insertion_sort (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Pred, typename _Distance>`  
`constexpr _Iter std::ranges::__detail::__find_if_not_n (_Iter __first, _Distance &__len, _Pred __pred)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__heap_select (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__inplace_stable_sort (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__insertion_sort (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__introselect (_Iter __first, _Iter __nth, _Iter __last, iter_difference_t<`  
`__Iter > __depth_limit, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__introsort_loop (_Iter __first, _Iter __last, unsigned __depth_limit, ←`  
`_Comp __comp)`
- `template<typename _Comp, typename _Proj>`  
`constexpr _Comp_proj< _Comp, _Proj > std::ranges::__detail::__make_comp_proj (_Comp &__comp, _Proj`  
`&__proj)`
- `template<typename _Pred, typename _Proj>`  
`constexpr _Pred_proj< _Pred, _Proj > std::ranges::__detail::__make_pred_proj (_Pred &__pred, _Proj &←`  
`__proj)`
- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::__detail::__merge_adaptive (_Iter __first, _Iter __middle, _Iter __last, iter_difference_t< _Iter`  
`> __len1, iter_difference_t< _Iter > __len2, _Pointer __buffer, _Comp __comp)`
- `template<typename _Iter, typename _Distance, typename _Pointer, typename _Comp>`  
`void std::ranges::__detail::__merge_adaptive_resize (_Iter __first, _Iter __middle, _Iter __last, _Distance ←`  
`__len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)`
- `template<typename _Iter, typename _Out, typename _Distance, typename _Comp>`  
`void std::ranges::__detail::__merge_sort_loop (_Iter __first, _Iter __last, _Out __result, _Distance __step←`  
`size, _Comp __comp)`
- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::__detail::__merge_sort_with_buffer (_Iter __first, _Iter __last, _Pointer __buffer, _Comp ←`  
`__comp)`
- `template<typename _Iter, typename _Distance, typename _Comp>`  
`constexpr void std::ranges::__detail::__merge_without_buffer (_Iter __first, _Iter __middle, _Iter __last, ←`  
`_Distance __len1, _Distance __len2, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__move_median_to_first (_Iter __result, _Iter __a, _Iter __b, _Iter __c,`  
`_Comp __comp)`
- `template<typename _Iter, typename _Out, typename _Comp>`  
`_Out std::ranges::__detail::__move_merge (_Iter __first1, _Iter __last1, _Iter __first2, _Iter __last2, _Out ←`  
`result, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2, typename _Out, typename _Comp>`  
`void std::ranges::__detail::__move_merge_adaptive (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 ←`  
`__last2, _Out __result, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2, typename _Iter3, typename _Comp>`  
`void std::ranges::__detail::__move_merge_adaptive_backward (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2,`  
`_Iter2 __last2, _Iter3 __result, _Comp __comp)`

- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__partial_sort (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__pop_heap (_Iter __first, _Iter __last, _Iter __result, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__push_heap (_Iter __first, iter_difference_t<_Iter> __holeIndex, iter_difference_t<_Iter> __topIndex, iter_value_t<_Iter> __value, _Comp __comp)`
- `template<typename _Iter1, typename _Iter2>`  
`_Iter1 std::ranges::__detail::__rotate_adaptive (_Iter1 __first, _Iter1 __middle, _Iter1 __last, iter_difference_t<_Iter1> __len1, iter_difference_t<_Iter1> __len2, _Iter2 __buffer, iter_difference_t<_Iter1> __buffer_size)`
- `template<typename _Iter, typename _Sent, typename _Pointer, typename _Pred, typename _Distance>`  
`constexpr subrange<_Iter> std::ranges::__detail::__stable_partition_adaptive (_Iter __first, _Sent __last, _Pred __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _Iter, typename _Pointer, typename _Comp>`  
`void std::ranges::__detail::__stable_sort_adaptive (_Iter __first, _Iter __middle, _Iter __last, _Pointer __buffer, _Comp __comp)`
- `template<typename _Iter, typename _Pointer, typename _Distance, typename _Comp>`  
`void std::ranges::__detail::__stable_sort_adaptive_resize (_Iter __first, _Iter __last, _Pointer __buffer, iter_difference_t<_Iter> __buffer_size, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__unguarded_insertion_sort (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr void std::ranges::__detail::__unguarded_linear_insert (_Iter __last, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr _Iter std::ranges::__detail::__unguarded_partition (_Iter __first, _Iter __last, _Iter __pivot, _Comp __comp)`
- `template<typename _Iter, typename _Comp>`  
`constexpr _Iter std::ranges::__detail::__unguarded_partition_pivot (_Iter __first, _Iter __last, _Comp __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::shift_left (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<_ForwardIterator>::difference_type __n)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::shift_right (_ForwardIterator __first, _ForwardIterator __last, typename iterator\_traits<_ForwardIterator>::difference_type __n)`

## Variables

- `constexpr int std::ranges::__detail::__sort_threshold`
- `constexpr __all_of_fn std::ranges::all_of`
- `constexpr __any_of_fn std::ranges::any_of`
- `constexpr __binary_search_fn std::ranges::binary_search`
- `constexpr __clamp_fn std::ranges::clamp`
- `constexpr __contains_fn std::ranges::contains`
- `constexpr __contains_subrange_fn std::ranges::contains_subrange`
- `constexpr __copy_if_fn std::ranges::copy_if`
- `constexpr __count_fn std::ranges::count`
- `constexpr __count_if_fn std::ranges::count_if`
- `constexpr __ends_with_fn std::ranges::ends_with`
- `constexpr __equal_range_fn std::ranges::equal_range`
- `constexpr __find_end_fn std::ranges::find_end`
- `constexpr __find_first_of_fn std::ranges::find_first_of`

- constexpr `__find_last_fn` **std::ranges::find\_last**
- constexpr `__find_last_if_fn` **std::ranges::find\_last\_if**
- constexpr `__find_last_if_not_fn` **std::ranges::find\_last\_if\_not**
- constexpr `__fold_left_fn` **std::ranges::fold\_left**
- constexpr `__fold_left_first_fn` **std::ranges::fold\_left\_first**
- constexpr `__fold_left_first_with_iter_fn` **std::ranges::fold\_left\_first\_with\_iter**
- constexpr `__fold_left_with_iter_fn` **std::ranges::fold\_left\_with\_iter**
- constexpr `__fold_right_fn` **std::ranges::fold\_right**
- constexpr `__fold_right_last_fn` **std::ranges::fold\_right\_last**
- constexpr `__for_each_fn` **std::ranges::for\_each**
- constexpr `__for_each_n_fn` **std::ranges::for\_each\_n**
- constexpr `__generate_fn` **std::ranges::generate**
- constexpr `__generate_n_fn` **std::ranges::generate\_n**
- constexpr `__includes_fn` **std::ranges::includes**
- constexpr `__inplace_merge_fn` **std::ranges::inplace\_merge**
- constexpr `__is_heap_fn` **std::ranges::is\_heap**
- constexpr `__is_heap_until_fn` **std::ranges::is\_heap\_until**
- constexpr `__is_partitioned_fn` **std::ranges::is\_partitioned**
- constexpr `__is_permutation_fn` **std::ranges::is\_permutation**
- constexpr `__is_sorted_fn` **std::ranges::is\_sorted**
- constexpr `__is_sorted_until_fn` **std::ranges::is\_sorted\_until**
- constexpr `__lexicographical_compare_fn` **std::ranges::lexicographical\_compare**
- constexpr `__lower_bound_fn` **std::ranges::lower\_bound**
- constexpr `__make_heap_fn` **std::ranges::make\_heap**
- constexpr `__max_fn` **std::ranges::max**
- constexpr `__max_element_fn` **std::ranges::max\_element**
- constexpr `__merge_fn` **std::ranges::merge**
- constexpr `__min_element_fn` **std::ranges::min\_element**
- constexpr `__minmax_fn` **std::ranges::minmax**
- constexpr `__minmax_element_fn` **std::ranges::minmax\_element**
- constexpr `__next_permutation_fn` **std::ranges::next\_permutation**
- constexpr `__none_of_fn` **std::ranges::none\_of**
- constexpr `__nth_element_fn` **std::ranges::nth\_element**
- constexpr `__partial_sort_fn` **std::ranges::partial\_sort**
- constexpr `__partial_sort_copy_fn` **std::ranges::partial\_sort\_copy**
- constexpr `__partition_fn` **std::ranges::partition**
- constexpr `__partition_copy_fn` **std::ranges::partition\_copy**
- constexpr `__partition_point_fn` **std::ranges::partition\_point**
- constexpr `__pop_heap_fn` **std::ranges::pop\_heap**
- constexpr `__prev_permutation_fn` **std::ranges::prev\_permutation**
- constexpr `__push_heap_fn` **std::ranges::push\_heap**
- constexpr `__remove_fn` **std::ranges::remove**
- constexpr `__remove_copy_fn` **std::ranges::remove\_copy**
- constexpr `__remove_copy_if_fn` **std::ranges::remove\_copy\_if**
- constexpr `__remove_if_fn` **std::ranges::remove\_if**
- constexpr `__replace_fn` **std::ranges::replace**
- constexpr `__replace_copy_fn` **std::ranges::replace\_copy**
- constexpr `__replace_copy_if_fn` **std::ranges::replace\_copy\_if**
- constexpr `__replace_if_fn` **std::ranges::replace\_if**
- constexpr `__reverse_fn` **std::ranges::reverse**
- constexpr `__reverse_copy_fn` **std::ranges::reverse\_copy**

- constexpr `__rotate_fn` **std::ranges::rotate**
- constexpr `__rotate_copy_fn` **std::ranges::rotate\_copy**
- constexpr `__sample_fn` **std::ranges::sample**
- constexpr `__search_n_fn` **std::ranges::search\_n**
- constexpr `__set_difference_fn` **std::ranges::set\_difference**
- constexpr `__set_intersection_fn` **std::ranges::set\_intersection**
- constexpr `__set_symmetric_difference_fn` **std::ranges::set\_symmetric\_difference**
- constexpr `__set_union_fn` **std::ranges::set\_union**
- constexpr `__shift_left_fn` **std::ranges::shift\_left**
- constexpr `__shift_right_fn` **std::ranges::shift\_right**
- constexpr `__shuffle_fn` **std::ranges::shuffle**
- constexpr `__sort_fn` **std::ranges::sort**
- constexpr `__sort_heap_fn` **std::ranges::sort\_heap**
- constexpr `__stable_partition_fn` **std::ranges::stable\_partition**
- constexpr `__stable_sort_fn` **std::ranges::stable\_sort**
- constexpr `__starts_with_fn` **std::ranges::starts\_with**
- constexpr `__swap_ranges_fn` **std::ranges::swap\_ranges**
- constexpr `__transform_fn` **std::ranges::transform**
- constexpr `__unique_fn` **std::ranges::unique**
- constexpr `__unique_copy_fn` **std::ranges::unique\_copy**
- constexpr `__upper_bound_fn` **std::ranges::upper\_bound**

### 7.125.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 7.126 ranges\_algobase.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Iter1, typename _Iter2>`  
using **std::ranges::copy\_backward\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::copy\_n\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::copy\_result**
- `template<typename _Iter1, typename _Iter2>`  
using **std::ranges::move\_backward\_result**
- `template<typename _Iter, typename _Out>`  
using **std::ranges::move\_result**

### Functions

- `template<bool _IsMove, typename _OutIter, typename _InIter>`  
constexpr void **std::ranges::\_\_detail::\_\_assign\_one** (\_OutIter &\_\_out, \_InIter &\_\_in)
- `template<bool _IsMove, input_iterator _Iter, sentinel_for<_Iter> _Sent, weakly_incrementable _Out>`  
constexpr `__conditional_t<_IsMove, move_result<_Iter, _Out>, copy_result<_Iter, _Out>>` **std::ranges::\_\_copy\_or\_move** (\_Iter \_\_first, \_Sent \_\_last, \_Out \_\_result)

- `template<bool _IsMove, bidirectional_iterator _Iter, sentinel_for< _Iter > _Sent, bidirectional_iterator _Out>  
constexpr __conditional_t< _IsMove, move_backward_result< _Iter, _Out >, copy_backward_result< _Iter, _Out  
> > std::ranges::__copy_or_move_backward ( _Iter __first, _Sent __last, _Out __result)`

## Variables

- `template<typename _Tp>  
constexpr bool std::ranges::__detail::__is_move_iterator`
- `template<typename _Iterator>  
constexpr bool std::ranges::__detail::__is_move_iterator< move_iterator< _Iterator > >`
- `template<typename _Tp>  
constexpr bool std::ranges::__detail::__is_normal_iterator`
- `template<typename _Iterator, typename _Container>  
constexpr bool std::ranges::__detail::__is_normal_iterator< gnu_cxx::__normal_iterator< _Iterator, _↵  
Container > >`
- `template<typename _Tp>  
constexpr bool std::ranges::__detail::__is_reverse_iterator`
- `template<typename _Iterator>  
constexpr bool std::ranges::__detail::__is_reverse_iterator< reverse_iterator< _Iterator > >`
- `constexpr __copy_fn std::ranges::copy`
- `constexpr __copy_backward_fn std::ranges::copy_backward`
- `constexpr __copy_n_fn std::ranges::copy_n`
- `constexpr __equal_fn std::ranges::equal`
- `constexpr __fill_fn std::ranges::fill`
- `constexpr __fill_n_fn std::ranges::fill_n`
- `constexpr __move_fn std::ranges::move`
- `constexpr __move_backward_fn std::ranges::move_backward`

### 7.126.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 7.127 ranges\_base.h File Reference

### Classes

- struct [std::ranges::dangling](#)
- struct [std::ranges::view\\_base](#)

### Namespaces

- namespace [std](#)

### Concepts

- concept [std::ranges::bidirectional\\_range](#)
- concept [std::ranges::borrowed\\_range](#)
- concept [std::ranges::common\\_range](#)
- concept [std::ranges::contiguous\\_range](#)
- concept [std::ranges::forward\\_range](#)
- concept [std::ranges::input\\_range](#)
- concept [std::ranges::output\\_range](#)
- concept [std::ranges::random\\_access\\_range](#)

- concept [std::ranges::range](#)
- concept [std::ranges::sized\\_range](#)
- concept [std::ranges::view](#)
- concept [std::ranges::viewable\\_range](#)

## Macros

- `#define _GLIBCXX26_RANGE_ALGO_DEF_VAL_T(_I, _P)`

## Typedefs

- `template<typename _Tp>`  
using **std::ranges::\_\_detail::\_\_make\_unsigned\_like\_t**
- `template<typename _Tp>`  
using **std::ranges::\_\_access::\_\_range\_iter\_t**
- `template<range _Range>`  
using **std::ranges::borrowed\_iterator\_t**
- `template<range _Range>`  
using **std::ranges::const\_iterator\_t**
- `template<range _Range>`  
using **std::ranges::const\_sentinel\_t**
- `template<typename _Tp>`  
using **std::ranges::iterator\_t**
- `template<range _Range>`  
using **std::ranges::range\_common\_reference\_t**
- `template<range _Range>`  
using **std::ranges::range\_const\_reference\_t**
- `template<range _Range>`  
using **std::ranges::range\_difference\_t**
- `template<range _Range>`  
using **std::ranges::range\_reference\_t**
- `template<range _Range>`  
using **std::ranges::range\_rvalue\_reference\_t**
- `template<sized_range _Range>`  
using **std::ranges::range\_size\_t**
- `template<range _Range>`  
using **std::ranges::range\_value\_t**
- `template<range _Range>`  
using **std::ranges::sentinel\_t**

## Functions

- `template<typename _Tp, typename _Up>`  
void **std::ranges::\_\_detail::\_\_is\_derived\_from\_view\_interface\_fn** (const \_Tp &, const [view\\_interface](#)<\_Up> &)
- `template<input_range _Range>`  
constexpr auto & **std::ranges::\_\_access::\_\_possibly\_const\_range** (\_Range &\_\_r) noexcept
- `template<__static_sized_range _Range>`  
constexpr range\_size\_t<\_Range> **std::ranges::\_\_static\_size** ()
- constexpr \_\_max\_size\_type **std::ranges::\_\_detail::\_\_to\_unsigned\_like** (\_\_max\_diff\_type \_\_t) noexcept
- constexpr \_\_max\_size\_type **std::ranges::\_\_detail::\_\_to\_unsigned\_like** (\_\_max\_size\_type \_\_t) noexcept
- `template<integral _Tp>`  
constexpr auto **std::ranges::\_\_detail::\_\_to\_unsigned\_like** (\_Tp \_\_t) noexcept

- void **std::ranges::\_\_access::end** ()=delete
- void **std::ranges::\_\_access::rbegin** ()=delete
- void **std::ranges::\_\_access::rend** ()=delete
- void **std::ranges::\_\_access::size** ()=delete

## Variables

- template<typename \_Tp>  
constexpr bool **std::ranges::\_\_detail::\_\_is\_initializer\_list**
- template<typename \_Tp>  
constexpr bool **std::ranges::\_\_detail::\_\_is\_initializer\_list**< initializer\_list< \_Tp > >
- constexpr \_\_advance\_fn **std::ranges::advance**
- constexpr ranges::\_\_access::\_Begin **std::ranges::begin**
- constexpr ranges::\_\_access::\_CBegin **std::ranges::cbegin**
- constexpr ranges::\_\_access::\_CData **std::ranges::cdata**
- constexpr ranges::\_\_access::\_CEnd **std::ranges::cend**
- constexpr ranges::\_\_access::\_CRBegin **std::ranges::crbegin**
- constexpr ranges::\_\_access::\_CREnd **std::ranges::crend**
- constexpr ranges::\_\_access::\_Data **std::ranges::data**
- template<typename>  
constexpr bool **std::ranges::disable\_sized\_range**
- constexpr \_\_distance\_fn **std::ranges::distance**
- constexpr ranges::\_\_access::\_Empty **std::ranges::empty**
- template<typename \_Tp>  
constexpr bool **std::ranges::enable\_borrowed\_range**
- template<typename \_Tp>  
constexpr bool **std::ranges::enable\_view**
- constexpr ranges::\_\_access::\_End **std::ranges::end**
- constexpr \_\_next\_fn **std::ranges::next**
- constexpr \_\_prev\_fn **std::ranges::prev**
- constexpr ranges::\_\_access::\_RBegin **std::ranges::rbegin**
- constexpr ranges::\_\_access::\_REnd **std::ranges::rend**
- constexpr ranges::\_\_access::\_Size **std::ranges::size**
- constexpr ranges::\_\_access::\_SSize **std::ranges::ssize**

### 7.127.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ranges>`.

### 7.127.2 Variable Documentation

#### **enable\_view**

```
template<typename _Tp>
bool std::ranges::enable_view [inline], [constexpr]
[range.view] The ranges::enable_view boolean.
```



## 7.128 ranges\_cmp.h File Reference

### Classes

- struct [std::ranges::equal\\_to](#)
- struct [std::ranges::greater](#)
- struct [std::ranges::greater\\_equal](#)
- struct [std::identity](#)
- struct [std::ranges::less](#)
- struct [std::ranges::less\\_equal](#)
- struct [std::ranges::not\\_equal\\_to](#)

### Namespaces

- namespace [std](#)

#### 7.128.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.129 ranges\_uninitialized.h File Reference

#### 7.129.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.130 ranges\_util.h File Reference

### Classes

- class [std::ranges::subrange<\\_It, \\_Sent, \\_Kind>](#)
- class [std::ranges::view\\_interface<\\_Derived>](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `template<range _Range>`  
using **std::ranges::borrowed\_subrange\_t**
- `template<typename _Iter1, typename _Iter2>`  
using **std::ranges::mismatch\_result**

### Enumerations

- enum class **subrange\_kind** : bool { **unsized** , **sized** }

### Functions

- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
constexpr auto **std::get** (const subrange<\_It, \_Sent, \_Kind> &\_\_r)
- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
constexpr auto **std::ranges::get** (const [subrange](#)<\_It, \_Sent, \_Kind> &\_\_r)

- `template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>`  
`constexpr auto std::ranges::get (subrange< _It, _Sent, _Kind > &&__r)`
- `template<input_or_output_iterator _It, sentinel_for< _It > _Sent>`  
`std::ranges::subrange (_It, _Sent) -> subrange< _It, _Sent >`
- `template<input_or_output_iterator _It, sentinel_for< _It > _Sent>`  
`std::ranges::subrange (_It, _Sent, __detail::__make_unsigned_like_t< iter_difference_t< _It > >) ->`  
`subrange< _It, _Sent, subrange_kind::sized >`
- `template<borrowed_range _Rng>`  
`std::ranges::subrange (_Rng &&) -> subrange< iterator_t< _Rng >, sentinel_t< _Rng >, (sized_range< _Rng`  
`>||sized_sentinel_for< sentinel_t< _Rng >, iterator_t< _Rng > >) ? subrange_kind::sized :subrange_kind::↵`  
`unsized >`
- `template<borrowed_range _Rng>`  
`std::ranges::subrange (_Rng &&, __detail::__make_unsigned_like_t< range_difference_t< _Rng > >) ->`  
`subrange< iterator_t< _Rng >, sentinel_t< _Rng >, subrange_kind::sized >`

## Variables

- `template<typename _It, typename _Sent, ranges::subrange_kind _Kind>`  
`constexpr bool std::__is_tuple_like_v< ranges::subrange< _It, _Sent, _Kind > >`
- `constexpr __adjacent_find_fn std::ranges::adjacent_find`
- `template<typename _It, typename _Sent, subrange_kind _Kind>`  
`constexpr bool std::ranges::enable_borrowed_range< subrange< _It, _Sent, _Kind > >`
- `constexpr __find_fn std::ranges::find`
- `constexpr __find_if_fn std::ranges::find_if`
- `constexpr __find_if_not_fn std::ranges::find_if_not`
- `constexpr __min_fn std::ranges::min`
- `constexpr __mismatch_fn std::ranges::mismatch`
- `constexpr __search_fn std::ranges::search`

### 7.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ranges>`.

## 7.131 `refwrap.h` File Reference

### Classes

- class [std::reference\\_wrapper](#)< \_Tp >

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _Tp>`  
`std::reference_wrapper (_Tp &) -> reference_wrapper< _Tp >`

### Variables

- `template<typename _Tp>`  
`constexpr bool std::__detail::__is_ref_wrapper`
- `template<typename _Tp>`  
`constexpr bool std::__detail::__is_ref_wrapper< reference_wrapper< _Tp > >`

### 7.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.132 regex.h File Reference

### Classes

- class `std::basic_regex< _Ch_type, _Rx_traits >`
- class `std::match_results< _Bi_iter, _Alloc >`
- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_traits< _Ch_type >`
- class `std::sub_match< _Biliter >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Typedefs

- typedef `match_results< const char * >` `std::cmatch`
- typedef `regex_iterator< const char * >` `std::cregex_iterator`
- typedef `regex_token_iterator< const char * >` `std::cregex_token_iterator`
- typedef `sub_match< const char * >` `std::csub_match`
- typedef `basic_regex< char >` `std::regex`
- typedef `match_results< string::const_iterator >` `std::smatch`
- typedef `regex_iterator< string::const_iterator >` `std::sregex_iterator`
- typedef `regex_token_iterator< string::const_iterator >` `std::sregex_token_iterator`
- typedef `sub_match< string::const_iterator >` `std::ssub_match`
- typedef `match_results< const wchar_t * >` `std::wcmatch`
- typedef `regex_iterator< const wchar_t * >` `std::wcregex_iterator`
- typedef `regex_token_iterator< const wchar_t * >` `std::wcregex_token_iterator`
- typedef `sub_match< const wchar_t * >` `std::wcs_sub_match`
- typedef `basic_regex< wchar_t >` `std::wregex`
- typedef `match_results< wstring::const_iterator >` `std::wsmatch`
- typedef `regex_iterator< wstring::const_iterator >` `std::wsregex_iterator`
- typedef `regex_token_iterator< wstring::const_iterator >` `std::wsregex_token_iterator`
- typedef `sub_match< wstring::const_iterator >` `std::wssub_match`

### Enumerations

- enum class `_RegexExecutorPolicy` : int { `_S_auto` , `_S_alternate` }

### Functions

- template<typename `_Biliter`, typename `_Alloc`, typename `_CharT`, typename `_TraitsT`>  
 bool `std::__detail::__regex_algo_impl` (`_Biliter` `_s`, `_Biliter` `_e`, `match_results< _Biliter, _Alloc >` &`_m`, const `basic_regex< _CharT, _TraitsT >` &`_re`, `regex_constants::match_flag_type` `_flags`, `_RegexExecutorPolicy` `_policy`, bool `_match_mode`)

- `template<typename _ForwardIterator>`  
**std::basic\_regex** (`_ForwardIterator`, `_ForwardIterator`, `regex_constants::syntax_option_type`={}) -> `basic_`  
`regex< typename iterator_traits< _ForwardIterator >::value_type >`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits>`  
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits>`  
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=`  
`regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type,`  
`_Rx_traits > &__, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-`  
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`  
`const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`  
`const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, match_results< const _Ch_type *, _Rx_traits > &__m, const regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits>`  
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, const regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)=delete`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

### 7.132.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.133 regex.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>`  
`_Out_iter std::__regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, const _Ch_type *__fmt, size_t __len, regex_constants::match_flag_type __flags)`

### 7.133.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.134 regex\_automaton.h File Reference

### Classes

- class `std::__detail::_StateSeq< _TraitsT >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

## Macros

- `#define _GLIBCXX_REGEX_STATE_LIMIT`

## Typedefs

- `template<typename _CharT>`  
  `using std::__detail::_Matcher`
- `typedef long std::__detail::_StatIdT`

## Enumerations

- `enum std::__detail::_Opcode : int {`  
  `_S_opcode_unknown , _S_opcode_alternative , _S_opcode_repeat , _S_opcode_backref ,`  
  `_S_opcode_line_begin_assertion , _S_opcode_line_end_assertion , _S_opcode_word_boundary , _S_↵`  
  `opcode_subexpr_lookahead ,`  
  `_S_opcode_subexpr_begin , _S_opcode_subexpr_end , _S_opcode_dummy , _S_opcode_match ,`  
  `_S_opcode_accept }`

## Variables

- `constexpr _StatIdT std::__detail::_S_invalid_state_id`

### 7.134.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.135 `regex_automaton.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### 7.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.136 `regex_compiler.h` File Reference

### Classes

- `struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >`
- `class std::__detail::_Compiler< _TraitsT >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### 7.136.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.137 regex\_compiler.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define __INSERT_REGEX_MATCHER(__func, ...)`

#### 7.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.138 regex\_constants.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::regex\\_constants](#)

### 5.1 Regular Expression Syntax Options

- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::\\_\\_multiline](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::\\_\\_polynomial](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::awk](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::basic](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::collate](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::ECMAScript](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::egrep](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::extended](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::grep](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::icase](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::multiline](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::nosubs](#)
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::operator&](#) ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) & [std::regex\\_constants::operator&=](#) ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::operator^](#) ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) & [std::regex\\_constants::operator^=](#) ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::operator|](#) ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) & [std::regex\\_constants::operator|=](#) ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b) noexcept
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::operator~](#) ([syntax\\_option\\_type](#) \_\_a) noexcept
- constexpr [syntax\\_option\\_type](#) [std::regex\\_constants::optimize](#)
- enum [std::regex\\_constants::syntax\\_option\\_type](#) : unsigned int {  
[\\_S\\_icase](#) , [\\_S\\_nosubs](#) , [\\_S\\_optimize](#) , [\\_S\\_collate](#) ,  
[\\_S\\_ECMAScript](#) , [\\_S\\_basic](#) , [\\_S\\_extended](#) , [\\_S\\_awk](#) ,  
[\\_S\\_grep](#) , [\\_S\\_egrep](#) , [\\_S\\_polynomial](#) , [\\_S\\_multiline](#) }

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- constexpr `match_flag_type` `std::regex_constants::format_default`
- constexpr `match_flag_type` `std::regex_constants::format_first_only`
- constexpr `match_flag_type` `std::regex_constants::format_no_copy`
- constexpr `match_flag_type` `std::regex_constants::format_sed`
- constexpr `match_flag_type` `std::regex_constants::match_any`
- constexpr `match_flag_type` `std::regex_constants::match_continuous`
- constexpr `match_flag_type` `std::regex_constants::match_default`
- enum `std::regex_constants::match_flag_type` : unsigned int {  
`_S_default` , `_S_not_bol` , `_S_not_eol` , `_S_not_bow` ,  
`_S_not_eow` , `_S_any` , `_S_not_null` , `_S_continuous` ,  
`_S_prev_avail` , `_S_sed` , `_S_no_copy` , `_S_first_only` ,  
`_S_match_flag_last` }
- constexpr `match_flag_type` `std::regex_constants::match_not_bol`
- constexpr `match_flag_type` `std::regex_constants::match_not_bow`
- constexpr `match_flag_type` `std::regex_constants::match_not_eol`
- constexpr `match_flag_type` `std::regex_constants::match_not_eow`
- constexpr `match_flag_type` `std::regex_constants::match_not_null`
- constexpr `match_flag_type` `std::regex_constants::match_prev_avail`
- constexpr `match_flag_type` `std::regex_constants::operator&` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & `std::regex_constants::operator&=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `std::regex_constants::operator^` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & `std::regex_constants::operator^=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `std::regex_constants::operator|` (`match_flag_type` \_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` & `std::regex_constants::operator|=` (`match_flag_type` &\_\_a, `match_flag_type` \_\_b) noexcept
- constexpr `match_flag_type` `std::regex_constants::operator~` (`match_flag_type` \_\_a) noexcept

### 7.138.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.139 regex\_error.h File Reference

### Classes

- class `std::regex_error`

### Namespaces

- namespace `std`
- namespace `std::regex_constants`



### 5.3 Error Types

- constexpr `error_type` `std::regex_constants::error_backref` (`_S_error_backref`)
- constexpr `error_type` `std::regex_constants::error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type` `std::regex_constants::error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type` `std::regex_constants::error_brace` (`_S_error_brace`)
- constexpr `error_type` `std::regex_constants::error_brack` (`_S_error_brack`)
- constexpr `error_type` `std::regex_constants::error_collate` (`_S_error_collate`)
- constexpr `error_type` `std::regex_constants::error_complexity` (`_S_error_complexity`)
- constexpr `error_type` `std::regex_constants::error_ctype` (`_S_error_ctype`)
- constexpr `error_type` `std::regex_constants::error_escape` (`_S_error_escape`)
- constexpr `error_type` `std::regex_constants::error_paren` (`_S_error_paren`)
- constexpr `error_type` `std::regex_constants::error_range` (`_S_error_range`)
- constexpr `error_type` `std::regex_constants::error_space` (`_S_error_space`)
- constexpr `error_type` `std::regex_constants::error_stack` (`_S_error_stack`)
- enum `std::regex_constants::error_type` {  
`_S_error_collate` , `_S_error_ctype` , `_S_error_escape` , `_S_error_backref` ,  
`_S_error_brack` , `_S_error_paren` , `_S_error_brace` , `_S_error_badbrace` ,  
`_S_error_range` , `_S_error_space` , `_S_error_badrepeat` , `_S_error_complexity` ,  
`_S_error_stack` , `_S_null` , `_S_grammar` }

#### 7.139.1 Detailed Description

Error and exception objects for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.140 `regex_executor.h` File Reference

### Classes

- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

#### 7.140.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.141 `regex_executor.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

## Enumerations

- enum `_ExecutorFrameOpcode` : unsigned char {  
    `_S_fopcode_next` , `_S_fopcode_fallback_next` , `_S_fopcode_rep_once_more` , `_S_fopcode_fallback_↵`  
    `rep_once_more` ,  
    `_S_fopcode_posix_alternative` , `_S_fopcode_merge_sol` , `_S_fopcode_restore_cur_results` , `_S_↵`  
    `fopcode_restore_rep_count` ,  
    `_S_fopcode_decrement_rep_count` }

### 7.141.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.142 `regex_scanner.h` File Reference

### Classes

- class `std::__detail::Scanner<_CharT>`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### 7.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.143 `regex_scanner.tcc` File Reference

### Namespaces

- namespace `std`
- namespace `std::__detail`

### 7.143.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 7.144 `requires_hosted.h` File Reference

### 7.144.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

## 7.145 `sat_arith.h` File Reference

### Namespaces

- namespace `std`

## Functions

- `template<typename _Tp>`  
`constexpr _Tp std::saturating_add (_Tp __x, _Tp __y) noexcept`
- `template<typename _Res, typename _Tp>`  
`constexpr _Res std::saturating_cast (_Tp __x) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp std::saturating_div (_Tp __x, _Tp __y) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp std::saturating_mul (_Tp __x, _Tp __y) noexcept`
- `template<typename _Tp>`  
`constexpr _Tp std::saturating_sub (_Tp __x, _Tp __y) noexcept`

### 7.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 7.146 semaphore\_base.h File Reference

### 7.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<semaphore>`.

## 7.147 bits/shared\_ptr.h File Reference

### Classes

- class `std::enable_shared_from_this< _Tp >`
- struct `std::hash< shared_ptr< _Tp > >`
- struct `std::owner_less< shared_ptr< _Tp > >`
- struct `std::owner_less< void >`
- struct `std::owner_less< weak_ptr< _Tp > >`
- class `std::shared_ptr< _Tp >`
- class `std::weak_ptr< _Tp >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Functions

- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Tp, typename _Del>`  
`std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >`
- `template<typename _Tp>`  
`std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >`
- `template<typename _Tp>`  
`std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >`

### 7.147.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.148 `experimental/bits/shared_ptr.h` File Reference

### Classes

- struct [std::hash< experimental::shared\\_ptr< \\_Tp > >](#)
- struct [std::experimental::fundamentals\\_v2::owner\\_less< shared\\_ptr< \\_Tp > >](#)
- struct [std::experimental::fundamentals\\_v2::owner\\_less< weak\\_ptr< \\_Tp > >](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Functions

- template<typename \_Tp>  
bool **std::experimental::atomic\_compare\_exchange\_strong** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w)
- template<typename \_Tp>  
bool **std::experimental::atomic\_compare\_exchange\_strong\_explicit** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w, [memory\\_order](#) \_\_success, [memory\\_order](#) \_\_failure)
- template<typename \_Tp>  
bool **std::experimental::atomic\_compare\_exchange\_weak** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w)
- template<typename \_Tp>  
bool **std::experimental::atomic\_compare\_exchange\_weak\_explicit** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w, [memory\\_order](#) \_\_success, [memory\\_order](#) \_\_failure)
- template<typename \_Tp>  
void **std::experimental::atomic\_exchange** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r)
- template<typename \_Tp>  
shared\_ptr< \_Tp > **std::experimental::atomic\_exchange\_explicit** (const shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r, [memory\\_order](#) \_\_mo)
- template<typename \_Tp>  
bool **std::experimental::atomic\_is\_lock\_free** (const shared\_ptr< \_Tp > \*\_\_p)
- template<typename \_Tp>  
shared\_ptr< \_Tp > **std::experimental::atomic\_load** (const shared\_ptr< \_Tp > \*\_\_p)
- template<typename \_Tp>  
shared\_ptr< \_Tp > **std::experimental::atomic\_load\_explicit** (const shared\_ptr< \_Tp > \*\_\_p, [memory\\_order](#) \_\_mo)
- template<typename \_Tp>  
void **std::experimental::atomic\_store** (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r)
- template<typename \_Tp>  
shared\_ptr< \_Tp > **std::experimental::atomic\_store\_explicit** (const shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r, [memory\\_order](#) \_\_mo)
- template<typename \_Tp, typename \_Tp1>  
shared\_ptr< \_Tp > **std::experimental::const\_pointer\_cast** (const shared\_ptr< \_Tp1 > &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1>  
shared\_ptr< \_Tp > **std::experimental::dynamic\_pointer\_cast** (const shared\_ptr< \_Tp1 > &\_\_r) noexcept

- `template<typename _Del, typename _Tp>`  
`_Del * std::experimental::get\_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp>`  
`std::basic\_ostream< _Ch, _Tr > & std::experimental::operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`
- `template<typename _Tp>`  
`bool std::experimental::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp1, typename _Tp2>`  
`bool std::experimental::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1>`  
`shared_ptr< _Tp > std::experimental::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp>`  
`void std::experimental::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`

- `template<typename _Tp>`  
`void std::experimental::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

## Variables

- `template<typename _Yp, typename _Tp>`  
`constexpr bool std::experimental::__sp_compatible_v`
- `template<typename _Tp, typename _Yp>`  
`constexpr bool std::experimental::__sp_is_constructible_v`

### 7.148.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/memory>`.

## 7.149 `shared_ptr_atomic.h` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_TSAN_MUTEX_DESTROY(X)`
- `#define _GLIBCXX_TSAN_MUTEX_LOCKED(X)`
- `#define _GLIBCXX_TSAN_MUTEX_POST_SIGNAL(X)`
- `#define _GLIBCXX_TSAN_MUTEX_POST_UNLOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_PRE_SIGNAL(X)`
- `#define _GLIBCXX_TSAN_MUTEX_PRE_UNLOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_TRY_LOCK(X)`
- `#define _GLIBCXX_TSAN_MUTEX_TRY_LOCK_FAILED(X)`

### Functions

- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)`
- `template<typename _Tp>`  
`bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`

- `template<typename _Tp>`  
`void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp>`  
`void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
  
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)`
- `template<typename _Tp>`  
`shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
  
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp>`  
`bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`

### 7.149.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.150 `shared_ptr_base.h` File Reference

### Classes

- struct `std::_Sp_ebo_helper<_Tp, false>`
- struct `std::_Sp_ebo_helper<_Tp, true>`
- class `std::bad_weak_ptr`
- struct `std::hash<__shared_ptr<_Tp, _Lp>>`

### Namespaces

- namespace `std`

### Enumerations

- enum `_Lock_policy`

### Functions

- template<typename \_Tp, \_Lock\_policy \_Lp = \_\_default\_lock\_policy, typename \_Alloc, typename... \_Args>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::allocate\_shared** (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- template<typename \_Tp, \_Lock\_policy \_Lp = \_\_default\_lock\_policy, typename... \_Args>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::make\_shared** (\_Args &&... \_\_args)
- template<typename \_Tp>  
\_Tp \* **std::shared\_ptr\_deref** (\_Tp \*\_\_p)
- void **std::throw\_bad\_weak\_ptr** ()
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::const\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::dynamic\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp, typename \_Up, \_Lock\_policy \_Lp>  
strong\_ordering **std::operator<=>** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Up, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
strong\_ordering **std::operator<=>** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
bool **std::operator==** (const \_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, nullptr\_t) noexcept
- template<typename \_Tp1, typename \_Tp2, \_Lock\_policy \_Lp>  
bool **std::operator==** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_a, const \_\_shared\_ptr<\_Tp2, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::reinterpret\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr<\_Tp, \_Lp> **std::static\_pointer\_cast** (const \_\_shared\_ptr<\_Tp1, \_Lp> &\_\_r) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
void **std::swap** (\_\_shared\_ptr<\_Tp, \_Lp> &\_\_a, \_\_shared\_ptr<\_Tp, \_Lp> &\_\_b) noexcept
- template<typename \_Tp, \_Lock\_policy \_Lp>  
void **std::swap** (\_\_weak\_ptr<\_Tp, \_Lp> &\_\_a, \_\_weak\_ptr<\_Tp, \_Lp> &\_\_b) noexcept

### Variables

- const \_Lock\_policy **std::\_\_default\_lock\_policy**



### 7.150.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.151 slice\_array.h File Reference

### Classes

- class [std::slice](#)
- class [std::slice\\_array<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### 7.151.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.152 specfun.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- [template<typename \\_Tp>  
\\_\\_gnu\\_cxx::\\_\\_promote<\\_Tp>::\\_\\_type \\_\\_gnu\\_cxx::airy\\_ai\(\\_Tp \\_\\_x\)](#)
- [float \\_\\_gnu\\_cxx::airy\\_aif\(float \\_\\_x\)](#)
- [long double \\_\\_gnu\\_cxx::airy\\_ail\(long double \\_\\_x\)](#)
- [template<typename \\_Tp>  
\\_\\_gnu\\_cxx::\\_\\_promote<\\_Tp>::\\_\\_type \\_\\_gnu\\_cxx::airy\\_bi\(\\_Tp \\_\\_x\)](#)
- [float \\_\\_gnu\\_cxx::airy\\_bif\(float \\_\\_x\)](#)
- [long double \\_\\_gnu\\_cxx::airy\\_bil\(long double \\_\\_x\)](#)
- [template<typename \\_Tp>  
\\_\\_gnu\\_cxx::\\_\\_promote<\\_Tp>::\\_\\_type std::assoc\\_laguerre\(unsigned int \\_\\_n, unsigned int \\_\\_m, \\_Tp \\_\\_x\)](#)
- [float std::assoc\\_laguerref\(unsigned int \\_\\_n, unsigned int \\_\\_m, float \\_\\_x\)](#)
- [long double std::assoc\\_laguerrel\(unsigned int \\_\\_n, unsigned int \\_\\_m, long double \\_\\_x\)](#)
- [template<typename \\_Tp>  
\\_\\_gnu\\_cxx::\\_\\_promote<\\_Tp>::\\_\\_type std::assoc\\_legendre\(unsigned int \\_\\_l, unsigned int \\_\\_m, \\_Tp \\_\\_x\)](#)
- [float std::assoc\\_legendref\(unsigned int \\_\\_l, unsigned int \\_\\_m, float \\_\\_x\)](#)
- [long double std::assoc\\_legendrel\(unsigned int \\_\\_l, unsigned int \\_\\_m, long double \\_\\_x\)](#)
- [template<typename \\_Tpa, typename \\_Tpb>  
\\_\\_gnu\\_cxx::\\_\\_promote\\_2<\\_Tpa, \\_Tpb>::\\_\\_type std::beta\(\\_Tpa \\_\\_a, \\_Tpb \\_\\_b\)](#)
- [float std::betaf\(float \\_\\_a, float \\_\\_b\)](#)
- [long double std::betal\(long double \\_\\_a, long double \\_\\_b\)](#)
- [template<typename \\_Tp>  
\\_\\_gnu\\_cxx::\\_\\_promote<\\_Tp>::\\_\\_type std::comp\\_ellint\\_1\(\\_Tp \\_\\_k\)](#)
- [float std::comp\\_ellint\\_1f\(float \\_\\_k\)](#)
- [long double std::comp\\_ellint\\_1l\(long double \\_\\_k\)](#)

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 ( _Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 ( _Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hypergf ( _Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`
- `long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_if (float __nu, float __x)`
- `long double std::cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_jf (float __nu, float __x)`
- `long double std::cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_kf (float __nu, float __x)`
- `long double std::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_neumannf (float __nu, float __x)`
- `long double std::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_1f (float __k, float __phi)`
- `long double std::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_2f (float __k, float __phi)`
- `long double std::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp>`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float std::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::expint ( _Tp __x)`
- `float std::expintf (float __x)`
- `long double std::expintl (long double __x)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)`
- `float std::hermitef (unsigned int __n, float __x)`
- `long double std::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)`

- long double [\\_\\_gnu\\_cxx::hypergl](#) (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::laguerre](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::laguerref](#) (unsigned int \_\_n, float \_\_x)
- long double [std::laguerrel](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::legendre](#) (unsigned int \_\_l, \_Tp \_\_x)
- float [std::legendref](#) (unsigned int \_\_l, float \_\_x)
- long double [std::legendrel](#) (unsigned int \_\_l, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::riemann\\_zeta](#) (\_Tp \_\_s)
- float [std::riemann\\_zetaf](#) (float \_\_s)
- long double [std::riemann\\_zetal](#) (long double \_\_s)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_bessel](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_besself](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_bessell](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float [std::sph\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

### 7.152.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>`.

## 7.153 sstream.tcc File Reference

### Namespaces

- namespace [std](#)

### 7.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.

## 7.154 std\_abs.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Tp>  
\_Tp [std::abs](#) (const [complex](#)< \_Tp > &)
- constexpr double [std::abs](#) (double \_\_x)
- constexpr float [std::abs](#) (float \_\_x)

- long **std::abs** (long \_\_i)
- constexpr long double **std::abs** (long double \_\_x)
- long long **std::abs** (long long \_\_x)

#### 7.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>` or `<cstdlib>`.

### 7.155 std\_function.h File Reference

#### Classes

- struct [std::\\_\\_is\\_location\\_invariant< \\_Tp >](#)
- class [std::\\_Function\\_base](#)
- class [std::bad\\_function\\_call](#)
- class [std::function< \\_Res\(\\_ArgTypes...\)>](#)

#### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

#### Typedefs

- `template<typename _Fn, typename _Op>`  
`using std::\_\_function\_guide\_t`

#### Enumerations

- enum [\\_Manager\\_operation](#) { [\\_\\_get\\_type\\_info](#) , [\\_\\_get\\_functor\\_ptr](#) , [\\_\\_clone\\_functor](#) , [\\_\\_destroy\\_functor](#) }

#### Functions

- `template<typename _Fn, typename _Signature = __function_guide_t< _Fn, decltype(&_Fn::operator())>>`  
`std::function ( _Fn ) -> function< _Signature >`
- `template<typename _Res, typename... _ArgTypes>`  
`std::function ( _Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>`
- `template<typename _Res, typename... _Args>`  
`bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept`

#### 7.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

### 7.156 std\_mutex.h File Reference

#### Classes

- struct [std::adopt\\_lock\\_t](#)
- struct [std::defer\\_lock\\_t](#)
- class [std::lock\\_guard< \\_Mutex >](#)

- class `std::mutex`
- struct `std::try_to_lock_t`

### Namespaces

- namespace `std`

### Variables

- constexpr `adopt_lock_t` `std::adopt_lock`
- constexpr `defer_lock_t` `std::defer_lock`
- constexpr `try_to_lock_t` `std::try_to_lock`

#### 7.156.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 7.157 `std_thread.h` File Reference

### Classes

- struct `std::hash< thread::id >`
- class `std::thread::id`
- class `std::thread`

### Namespaces

- namespace `std`
- namespace `std::this_thread`

### Functions

- `thread::id` `std::this_thread::get_id ()` noexcept
- void `std::this_thread::yield ()` noexcept

#### 7.157.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<thread>`.

## 7.158 `stdexcept_except.h` File Reference

### Classes

- class `std::domain_error`
- class `std::invalid_argument`
- class `std::length_error`
- class `std::logic_error`
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::runtime_error`
- class `std::underflow_error`

## Namespaces

- namespace [std](#)

## Typedefs

- typedef [basic\\_string](#)< char > **std::\_\_sso\_string**

### 7.158.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stdexcept>`.

## 7.159 `stdexcept_throw.h` File Reference

### 7.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stdexcept>`.

## 7.160 `stdexcept_throwdef.h` File Reference

### 7.160.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stdexcept>`.

## 7.161 `stdexcept_throwfwd.h` File Reference

## Namespaces

- namespace [std](#)

## Functions

- void **std::\_\_throw\_domain\_error** (const char \*)
- void **std::\_\_throw\_invalid\_argument** (const char \*)
- void **std::\_\_throw\_length\_error** (const char \*)
- void **std::\_\_throw\_logic\_error** (const char \*)
- void **std::\_\_throw\_out\_of\_range** (const char \*)
- void **std::\_\_throw\_out\_of\_range\_fmt** (const char \*,...)
- void **std::\_\_throw\_overflow\_error** (const char \*)
- void **std::\_\_throw\_runtime\_error** (const char \*)
- void **std::\_\_throw\_underflow\_error** (const char \*)

### 7.161.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stdexcept>`.

## 7.162 `stl_algo.h` File Reference

## Namespaces

- namespace [std](#)

## Enumerations

- enum { **`_S_chunk_size`** }

## Functions

- template<typename `_ForwardIterator`, typename `_BinaryPredicate`>  
constexpr `_ForwardIterator` **`std::adjacent_find`** (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `_BinaryPredicate` \_\_binary\_pred)
- template<typename `_RandomAccessIterator`, typename `_Distance`, typename `_Compare`>  
constexpr void **`std::chunk_insertion_sort`** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Distance` \_\_chunk\_size, `_Compare` \_\_comp)
- template<typename `_ForwardIterator`, typename `_Tp`, typename `_Compare`>  
constexpr pair< `_ForwardIterator`, `_ForwardIterator` > **`std::equal_range`** (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, const `_Tp` &\_\_val, `_Compare` \_\_comp)
- template<typename `_BidirectionalIterator1`, typename `_BidirectionalIterator2`, typename `_BinaryPredicate`>  
constexpr `_BidirectionalIterator1` **`std::find_end`** (`_BidirectionalIterator1` \_\_first1, `_BidirectionalIterator1` \_\_last1, `_BidirectionalIterator2` \_\_first2, `_BidirectionalIterator2` \_\_last2, `_BinaryPredicate` \_\_comp)
- template<typename `_ForwardIterator1`, typename `_ForwardIterator2`, typename `_BinaryPredicate`>  
constexpr `_ForwardIterator1` **`std::find_end`** (`_ForwardIterator1` \_\_first1, `_ForwardIterator1` \_\_last1, `_ForwardIterator2` \_\_first2, `_ForwardIterator2` \_\_last2, `_BinaryPredicate` \_\_comp)
- template<typename `_InputIterator`, typename `_Predicate`>  
constexpr `_InputIterator` **`std::find_if_not`** (`_InputIterator` \_\_first, `_InputIterator` \_\_last, `_Predicate` \_\_pred)
- template<typename `_InputIterator`, typename `_Predicate`, typename `_Distance`>  
constexpr `_InputIterator` **`std::find_if_not_n`** (`_InputIterator` \_\_first, `_Distance` &\_\_len, `_Predicate` \_\_pred)
- template<typename `_EuclideanRingElement`>  
constexpr `_EuclideanRingElement` **`std::gcd`** (`_EuclideanRingElement` \_\_m, `_EuclideanRingElement` \_\_n)
- template<typename `_IntType`, typename `_UniformRandomBitGenerator`>  
pair< `_IntType`, `_IntType` > **`std::gen_two_uniform_ints`** (`_IntType` \_\_b0, `_IntType` \_\_b1, `_UniformRandomBitGenerator` &&\_\_g)
- template<typename `_InputIterator1`, typename `_InputIterator2`, typename `_Compare`>  
constexpr bool **`std::includes`** (`_InputIterator1` \_\_first1, `_InputIterator1` \_\_last1, `_InputIterator2` \_\_first2, `_InputIterator2` \_\_last2, `_Compare` \_\_comp)
- template<typename `_BidirectionalIterator`, typename `_Compare`>  
constexpr void **`std::inplace_merge`** (`_BidirectionalIterator` \_\_first, `_BidirectionalIterator` \_\_middle, `_BidirectionalIterator` \_\_last, `_Compare` \_\_comp)
- template<typename `_RandomAccessIterator`, typename `_Compare`>  
constexpr void **`std::inplace_stable_sort`** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Compare` \_\_comp)
- template<typename `_ForwardIterator1`, typename `_ForwardIterator2`, typename `_BinaryPredicate`>  
constexpr bool **`std::is_permutation`** (`_ForwardIterator1` \_\_first1, `_ForwardIterator1` \_\_last1, `_ForwardIterator2` \_\_first2, `_ForwardIterator2` \_\_last2, `_BinaryPredicate` \_\_pred)
- template<typename `_ForwardIterator`, typename `_Compare`>  
constexpr `_ForwardIterator` **`std::is_sorted_until`** (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `_Compare` \_\_comp)
- template<typename `_ForwardIterator`, typename `_Compare`>  
constexpr `_ForwardIterator` **`std::max_element`** (`_ForwardIterator` \_\_first, `_ForwardIterator` \_\_last, `_Compare` \_\_comp)
- template<typename `_InputIterator1`, typename `_InputIterator2`, typename `_OutputIterator`, typename `_Compare`>  
constexpr `_OutputIterator` **`std::merge`** (`_InputIterator1` \_\_first1, `_InputIterator1` \_\_last1, `_InputIterator2` \_\_first2, `_InputIterator2` \_\_last2, `_OutputIterator` \_\_result, `_Compare` \_\_comp)

- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void std::\_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>`  
`void std::\_\_merge\_adaptive\_resize (_BidirectionalIterator __first, _BidirectionalIterator __middle, \_↔ _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>`  
`void std::\_\_merge\_sort\_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccess↔ _AccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void std::\_\_merge\_sort\_with\_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare>`  
`constexpr void std::\_\_merge\_without\_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, \_↔ _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::\_\_min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::\_\_minmax\_element (_ForwardIterator __first, \_↔ _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _Compare>`  
`constexpr void std::\_\_move\_median\_to\_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, \_↔ _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::\_\_move\_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, \_↔ _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`void std::\_\_move\_merge\_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, \_↔ _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>`  
`void std::\_\_move\_merge\_adaptive\_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, \_↔ _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::\_\_next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Predicate>`  
`constexpr _BidirectionalIterator std::\_\_partition (_BidirectionalIterator __first, _BidirectionalIterator __last, \_↔ _Predicate __pred, bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::\_\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::\_\_prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::\_\_remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator std::\_\_replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`



- `template<typename _BidirectionalIterator>`  
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator std::rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, ↵  
_BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator  
__last, forward\_iterator\_tag)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::rotate (_RandomAccessIterator __first, _RandomAccessIterator __↵  
middle, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>`  
`_BidirectionalIterator1 std::rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵  
_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance  
__buffer_size)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵  
Generator>`  
`_OutputIterator std::sample (_ForwardIterator __first, _ForwardIterator __last, forward\_iterator\_tag, _Output↵  
Iterator __out, _Cat, _Size __n, _UniformRandomBitGenerator &&g)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>`  
`_RandomAccessIterator std::sample (_InputIterator __first, _InputIterator __last, input\_iterator\_tag, ↵  
_RandomAccessIterator __out, random\_access\_iterator\_tag, _Size __n, _UniformRandomBitGenerator &&g)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count,  
_UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer ↵  
__count, _UnaryPredicate __unary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _RandomAccessIterator std::search_n_aux (_RandomAccessIterator __first, _RandomAccessIterator __last, ↵  
_Integer __count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2  
__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵  
Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, ↵  
_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 ↵  
__first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate  
__pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>`  
`constexpr _ForwardIterator std::stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, ↵  
_Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare  
__comp)`

- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>`  
`void std::stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __middle, ↵`  
`_RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>`  
`void std::stable_sort_adaptive_resize (_RandomAccessIterator __first, _RandomAccessIterator __last, ↵`  
`_Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__result, _BinaryPredicate __binary_pred, forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator ↵`  
`__result, _BinaryPredicate __binary_pred, input\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`_ForwardIterator std::unique_copy_1 (_InputIterator __first, _InputIterator __last, _ForwardIterator __result,`  
`_BinaryPredicate __binary_pred, __true_type)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique_copy_1 (_InputIterator __first, _InputIterator __last, _OutputIterator ↵`  
`__result, _BinaryPredicate __binary_pred, __false_type)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`&__val, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare`  
`__comp)`
- `template<typename _Tp>`  
`constexpr const _Tp &std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)`
- `template<typename _Tp, typename _Compare>`  
`constexpr const _Tp &std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`  
`_Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last,`  
`const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator ↵`  
`__last, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function>`  
`constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _InputIterator, typename _Size, typename _Function>`  
`constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator>`  
`constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare>`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator>`  
`constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::max (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::max (initializer\_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::min (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr _Tp std::min (initializer\_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare>`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer\_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare>`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

- `template<typename _BidirectionalIterator>`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>`  
`constexpr pair<_OutputIterator1, _OutputIterator2> std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare>`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator>`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate>`  
`constexpr _OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>`  
`constexpr _OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp>`  
`constexpr void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator>`  
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>`  
`_SampleIterator std::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`constexpr _ForwardIterator std::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp>`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber↵`  
`Generator &&__g)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate ↵`  
`__pred)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare ↵`  
`comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`  
`_UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate ↵`  
`binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &↵`  
`val, _Compare __comp)`

### 7.162.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.



## 7.163 `std_algobase.h` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_ADVANCE(P, N)`
- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_TO_ADDR(P)`

### Functions

- `template<bool _IsMove, typename _OutIter, typename _InIter>`  
`constexpr void std::__assign_one (_OutIter &__out, _InIter &__in)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::__copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::__copy_move_a (_II, _II, const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _OIte, _OSeq, _OCat > std::__copy_move_a (const \_\_gnu\_debug::Safe\_iterator< _IIte, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _IIte, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::__copy_move_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`\_\_Deque\_iterator< _OTp, _OTp &, _OTp * > std::__copy_move_a1 (\_\_Deque\_iterator< _ITp, _IRef, _IPtr > __first, \_\_Deque\_iterator< _ITp, _IRef, _IPtr > __last, \_\_Deque\_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI std::__copy_move_a1 (\_\_Deque\_iterator< _Tp, _Ref, _Ptr > __first, \_\_Deque\_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`\_\_gnu\_cxx::enable\_if< \_\_is\_any\_random\_access\_iter< _II >::__value, \_\_Deque\_iterator< _Tp, _Tp &, _Tp * > >::__type std::__copy_move_a1 (_II __first, _II __last, \_\_Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::__copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT>`  
`\_\_gnu\_cxx::enable\_if< \_\_is\_char< _CharT >::__value, ostreambuf\_iterator< _CharT, char\_traits< _CharT > > >::__type std::__copy_move_a2 (_CharT *, _CharT *, ostreambuf\_iterator< _CharT, char\_traits< _CharT > >)`
- `template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>`  
`constexpr _OutIter std::__copy_move_a2 (_InIter __first, _Sent __last, _OutIter __result)`
- `template<bool _IsMove, typename _CharT>`  
`\_\_gnu\_cxx::enable\_if< \_\_is\_char< _CharT >::__value, ostreambuf\_iterator< _CharT, char\_traits< _CharT > > >::__type std::__copy_move_a2 (const _CharT *, const _CharT *, ostreambuf\_iterator< _CharT, char\_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT>`  
`\_\_gnu\_cxx::enable\_if< \_\_is\_char< _CharT >::__value, \_\_Deque\_iterator< _CharT, _CharT &, _CharT * > >::__type std::__copy_move_a2 (istreambuf\_iterator< _CharT, char\_traits< _CharT > > __first, istreambuf\_iterator< _CharT, char\_traits< _CharT > > __last, \_\_Deque\_iterator< _CharT, _CharT &, \_\_CharT * > __result)`



- `template<bool _IsMove, typename _CharT>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT`  
`> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI>`  
`constexpr _OI std::__copy_move_backward_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr __gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > std::__copy_move_backward_a (_II, _II, const`  
`::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>`  
`constexpr ::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > std::__copy_move_backward_a (const`  
`::__gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat`  
`> &, const ::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::__copy_move_backward_a (const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &,`  
`const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std::__copy_move_backward_a1 (::Deque_iterator< _ITp, _IRef,`  
`_IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp *`  
`> __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>`  
`_OI std::__copy_move_backward_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 std::__copy_move_backward_a1 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _II, typename _Tp>`  
`__gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp *`  
`> >::__type std::__copy_move_backward_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp *`  
`> __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2>`  
`constexpr _BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator>`  
`constexpr _OutputIterator std::__copy_n_a (_InputIterator __first, _Size __n, _OutputIterator __result, bool)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ::Deque_iterator< _CharT, _CharT &, _CharT *`  
`> >::__type std::__copy_n_a (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size,`  
`::Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)`
- `template<typename _CharT, typename _Size>`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_n_a (istreambuf_iterator<`  
`_CharT, char_traits< _CharT > >, _Size, _CharT *, bool)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr iterator_traits< _InputIterator >::difference_type std::__count_if (_InputIterator __first, _InputIterator`  
`__last, _Predicate __pred)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _BinaryPredicate>`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary,`  
`_pred)`
- `template<typename _II1, typename _II2>`  
`constexpr bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__equal_aux (_II1, _II1, const ::__gnu_debug::Safe_iterator< _II2, _Seq2, _Cat2 > &)`

- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::equal_aux (const ::gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const ::gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, _I2)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::equal_aux (const ::gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const ::gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const ::gnu\_debug::Safe\_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _I1>`  
`gnu\_cxx::enable\_if< is\_any\_random\_access\_iter< _I1 >::value, bool >::type std::equal_aux1 (:: Deque\_iterator< _Tp, _Ref, _Ptr > __first1, :: Deque\_iterator< _Tp, _Ref, _Ptr > __last1, _I1 __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::equal_aux1 (:: Deque\_iterator< _Tp1, _Ref1, _Ptr1 > __first1, :: Deque\_iterator< _Tp1, _Ref1, _Ptr1 > __last1, :: Deque\_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _I1, typename _Tp, typename _Ref, typename _Ptr>`  
`gnu\_cxx::enable\_if< is\_any\_random\_access\_iter< _I1 >::value, bool >::type std::equal_aux1 (_I1 __first1, _I1 __last1, :: Deque\_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::equal_aux1 (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _FIter, typename _Tp>`  
`constexpr void std::fill_a (_FIter __first, _FIter __last, const _Tp &__value)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Tp>`  
`constexpr void std::fill_a (const ::gnu\_debug::Safe\_iterator< _Lte, _Seq, _Cat > &, const ::gnu\_debug::Safe\_iterator< _Lte, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Lte, typename _Cont, typename _Tp>`  
`constexpr void std::fill_a1 (:: gnu\_cxx::normal\_iterator< _Lte, _Cont > __first, :: gnu\_cxx::normal\_iterator< _Lte, _Cont > __last, const _Tp &__value)`
- `constexpr void std::fill_a1 (:: Bit\_iterator, :: Bit\_iterator, const bool &)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::fill_a1 (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Up, typename _Tp>`  
`constexpr gnu\_cxx::enable\_if< is\_byte< _Up >::value &&(are\_same< _Up, _Tp >::value || memcpable\_integer< _Tp >::width), void >::type std::fill_a1 (_Up *__first, _Up *__last, const _Tp &__x)`
- `template<typename _Tp, typename _VTp>`  
`void std::fill_a1 (const ::Deque\_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque\_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::input\_iterator\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::output\_iterator\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::random\_access\_iterator\_tag)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Size, typename _Tp>`  
`constexpr ::gnu\_debug::Safe\_iterator< _Lte, _Seq, _Cat > std::fill_n_a (const ::gnu\_debug::Safe\_iterator< _Lte, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input\_iterator\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp>`  
`constexpr _OutputIterator std::fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Iterator, typename _Predicate>`  
`constexpr _Iterator std::find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`

- `template<typename _I1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _Iter2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (_I1, _I1, const ::__gnu_debug::Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::Safe_iterator< _Iter2, _Seq2, _Cat2 > &)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _I2, _I2)`
- `template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::__lexicographical_compare_aux (const ::__gnu_debug::Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::Safe_iterator< _Iter2, _Seq2, _Cat2 > &)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>`  
`bool std::__lexicographical_compare_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 * __first2, _Tp2 * __last2)`
- `template<typename _I1, typename _I2>`  
`constexpr bool std::__lexicographical_compare_aux1 (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>`  
`bool std::__lexicographical_compare_aux1 (_Tp1 * __first1, _Tp1 * __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)`
- `template<typename _I1, typename _I2, typename _Compare>`  
`constexpr bool std::__lexicographical_compare_impl (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _Tp>`  
`constexpr _Tp std::__lg (_Tp __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare>`  
`constexpr _ForwardIterator std::__lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Tp, typename _Up>`  
`constexpr int std::__memcmp (const _Tp * __first1, const _Up * __first2, size_t __num)`
- `template<typename _Tp>`  
`constexpr auto std::__min_cmp (_Tp __x, _Tp __y)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::__mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::__mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr _ForwardIterator std::__remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::__search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `constexpr long long std::__size_to_integer (double __n)`
- `constexpr long long std::__size_to_integer (float __n)`
- `constexpr int std::__size_to_integer (int __n)`
- `constexpr long std::__size_to_integer (long __n)`
- `constexpr long long std::__size_to_integer (long double __n)`

- constexpr long long **std::size\_to\_integer** (long long \_\_n)
- constexpr unsigned **std::size\_to\_integer** (unsigned \_\_n)
- constexpr unsigned long **std::size\_to\_integer** (unsigned long \_\_n)
- constexpr unsigned long long **std::size\_to\_integer** (unsigned long long \_\_n)
- template<typename \_II, typename \_OI>  
constexpr **std::copy** (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- template<typename \_BI1, typename \_BI2>  
constexpr **std::copy\_backward** (\_BI1 \_\_first, \_BI1 \_\_last, \_BI2 \_\_result)
- template<typename \_II1, typename \_II2>  
constexpr bool **std::equal** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2)
- template<typename \_II1, typename \_II2>  
constexpr bool **std::equal** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2, \_II2 \_\_last2)
- template<typename \_IIter1, typename \_IIter2, typename \_BinaryPredicate>  
constexpr bool **std::equal** (\_IIter1 \_\_first1, \_IIter1 \_\_last1, \_IIter2 \_\_first2, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_IIter1, typename \_IIter2, typename \_BinaryPredicate>  
constexpr bool **std::equal** (\_IIter1 \_\_first1, \_IIter1 \_\_last1, \_IIter2 \_\_first2, \_IIter2 \_\_last2, \_BinaryPredicate \_\_binary\_pred) ↵
- template<typename \_ForwardIterator, typename \_Tp>  
constexpr void **std::fill** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_value)
- template<typename \_OI, typename \_Size, typename \_Tp>  
constexpr **std::fill\_n** (\_OI \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2>  
constexpr bool **std::is\_permutation** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2 \_\_first2) ↵
- template<typename \_ForwardIterator1, typename \_ForwardIterator2>  
constexpr void **std::iter\_swap** (\_ForwardIterator1 \_\_a, \_ForwardIterator2 \_\_b)
- template<typename \_II1, typename \_II2>  
constexpr bool **std::lexicographical\_compare** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2, \_II2 \_\_last2)
- template<typename \_II1, typename \_II2, typename \_Compare>  
constexpr bool **std::lexicographical\_compare** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2, \_II2 \_\_last2, \_Compare \_\_comp)
- template<typename \_InputIter1, typename \_InputIter2>  
constexpr auto **std::lexicographical\_compare\_three\_way** (\_InputIter1 \_\_first1, \_InputIter1 \_\_last1, \_InputIter2 \_\_first2, \_InputIter2 \_\_last2)
- template<typename \_InputIter1, typename \_InputIter2, typename \_Comp>  
constexpr auto **std::lexicographical\_compare\_three\_way** (\_InputIter1 \_\_first1, \_InputIter1 \_\_last1, \_InputIter2 \_\_first2, \_InputIter2 \_\_last2, \_Comp \_\_comp) -> decltype(\_\_comp(\*\_\_first1, \*\_\_first2))
- template<typename \_ForwardIterator, typename \_Tp>  
constexpr \_ForwardIterator **std::lower\_bound** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val)
- template<typename \_Tp>  
constexpr const \_Tp & **std::max** (const \_Tp &\_\_a, const \_Tp &\_\_b)
- template<typename \_Tp, typename \_Compare>  
constexpr const \_Tp & **std::max** (const \_Tp &\_\_a, const \_Tp &\_\_b, \_Compare \_\_comp)
- template<typename \_Tp>  
constexpr const \_Tp & **std::min** (const \_Tp &\_\_a, const \_Tp &\_\_b)
- template<typename \_Tp, typename \_Compare>  
constexpr const \_Tp & **std::min** (const \_Tp &\_\_a, const \_Tp &\_\_b, \_Compare \_\_comp)
- template<typename \_InputIterator1, typename \_InputIterator2>  
constexpr pair< \_InputIterator1, \_InputIterator2 > **std::mismatch** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2)
- template<typename \_InputIterator1, typename \_InputIterator2, typename \_BinaryPredicate>  
constexpr pair< \_InputIterator1, \_InputIterator2 > **std::mismatch** (\_InputIterator1 \_\_first1, \_InputIterator1 \_\_last1, \_InputIterator2 \_\_first2, \_BinaryPredicate \_\_binary\_pred)

- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI>`  
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2>`  
`constexpr _BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2>`  
`constexpr _ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

### 7.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 7.164 `std_bvector.h` File Reference

### Classes

- struct [std::hash<::vector< bool, \\_Alloc > >](#)
- class [std::vector< bool, \\_Alloc >](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_ALWAYS\_INLINE`

### Typedefs

- typedef unsigned long [std::\\_Bit\\_type](#)

### Enumerations

- enum { [\\_S\\_word\\_bit](#) }

### Functions

- `constexpr void std::\_fill\_a1 (::_Bit_iterator, ::_Bit_iterator, const bool &)`
- `constexpr void std::\_fill\_bvector (_Bit_type * __v, unsigned int __first, unsigned int __last, bool __x) noexcept`
- `constexpr void std::\_fill\_bvector\_n (_Bit_type *, size_t, bool) noexcept`

### 7.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 7.165 `std_construct.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp, typename... _Args>`  
`constexpr void std::\_Construct (_Tp *__p, _Args &&... __args)`
- `template<typename _T1>`  
`constexpr void std::\_Construct\_novalue (_T1 *__p)`
- `template<typename _ForwardIterator>`  
`constexpr void std::\_Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp>`  
`constexpr void std::\_Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::\_Destroy\_n (_ForwardIterator __first, _Size __count)`
- `template<typename _Tp, typename... _Args>`  
`constexpr _Tp * std::construct\_at (_Tp *__location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval< _Args >().)))`
- `template<typename _ForwardIterator>`  
`constexpr void std::destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp>`  
`constexpr void std::destroy\_at (_Tp *__location)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::destroy\_n (_ForwardIterator __first, _Size __count)`
- `template<typename _Tp>`  
`const _Tp * std::start\_lifetime\_as (const void *__p) noexcept`
- `template<typename _Tp>`  
`const volatile _Tp * std::start\_lifetime\_as (const volatile void *__p) noexcept`
- `template<typename _Tp>`  
`_Tp * std::start\_lifetime\_as (void *__p) noexcept`
- `template<typename _Tp>`  
`volatile _Tp * std::start\_lifetime\_as (volatile void *__p) noexcept`
- `template<typename _Tp>`  
`const _Tp * std::start\_lifetime\_as\_array (const void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`const volatile _Tp * std::start\_lifetime\_as\_array (const volatile void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`_Tp * std::start\_lifetime\_as\_array (void *__p, size_t __n) noexcept`
- `template<typename _Tp>`  
`volatile _Tp * std::start\_lifetime\_as\_array (volatile void *__p, size_t __n) noexcept`

### 7.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.166 `std_deque.h` File Reference

### Classes

- class [std::\\_Deque\\_base](#)< \_Tp, \_Alloc >
- struct [std::\\_Deque\\_iterator](#)< \_Tp, \_Ref, \_Ptr >
- class [std::deque](#)< \_Tp, \_Alloc >

## Namespaces

- namespace `std`

## Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

## Functions

- `constexpr size_t std::__deque_buf_size (size_t __size)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>> std::deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque<_ValT, _Allocator>`
- `template<typename _Tp, typename _Alloc> __detail::__synth3way_t<_Tp> std::operator<=> (const deque<_Tp, _Alloc> &__x, const deque<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> bool std::operator== (const deque<_Tp, _Alloc> &__x, const deque<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc> void std::swap (deque<_Tp, _Alloc> &__x, deque<_Tp, _Alloc> &__y) noexcept(/*conditional */)`

### 7.166.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

### 7.166.2 Macro Definition Documentation

#### `_GLIBCXX_DEQUE_BUF_SIZE`

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

#### Parameters

|                     |                         |
|---------------------|-------------------------|
| <code>__size</code> | The size of an element. |
|---------------------|-------------------------|

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

## 7.167 `std_function.h` File Reference

### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result>`
- class `std::binary_negate<_Predicate>`
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`

- class `std::const_mem_fun_ref_t<_Ret, _Tp>`
- class `std::const_mem_fun_t<_Ret, _Tp>`
- struct `std::divides<_Tp>`
- struct `std::divides<void>`
- struct `std::equal_to<_Tp>`
- struct `std::equal_to<void>`
- struct `std::greater<_Tp>`
- struct `std::greater<void>`
- struct `std::greater_equal<_Tp>`
- struct `std::greater_equal<void>`
- struct `std::less<_Tp>`
- struct `std::less<void>`
- struct `std::less_equal<_Tp>`
- struct `std::less_equal<void>`
- struct `std::logical_and<_Tp>`
- struct `std::logical_and<void>`
- struct `std::logical_not<_Tp>`
- struct `std::logical_not<void>`
- struct `std::logical_or<_Tp>`
- struct `std::logical_or<void>`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun_ref_t<_Ret, _Tp>`
- class `std::mem_fun_t<_Ret, _Tp>`
- struct `std::minus<_Tp>`
- struct `std::minus<void>`
- struct `std::modulus<_Tp>`
- struct `std::modulus<void>`
- struct `std::multiplies<_Tp>`
- struct `std::multiplies<void>`
- struct `std::negate<_Tp>`
- struct `std::negate<void>`
- struct `std::not_equal_to<_Tp>`
- struct `std::not_equal_to<void>`
- struct `std::plus<_Tp>`
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`
- class `std::pointer_to_unary_function<_Arg, _Result>`
- struct `std::unary_function<_Arg, _Result>`
- class `std::unary_negate<_Predicate>`

## Namespaces

- namespace `std`

## Typedefs

- template<typename \_Func, typename \_SfinaeType>  
using `std::__has_is_transparent_t`



## Functions

- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_t< _Ret, _Tp > std::mem_fun ( _Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_t< _Ret, _Tp > std::mem_fun ( _Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun ( _Ret(_Tp::* __f)( _Arg) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun ( _Ret(_Tp::* __f)( _Arg))`
- `template<typename _Ret, typename _Tp>`  
`const_mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref ( _Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp>`  
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref ( _Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref ( _Ret(_Tp::* __f)( _Arg) const)`
- `template<typename _Ret, typename _Tp, typename _Arg>`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref ( _Ret(_Tp::* __f)( _Arg))`
- `template<typename _Predicate>`  
`constexpr unary_negate< _Predicate > std::not1 (const _Predicate & __pred)`
- `template<typename _Predicate>`  
`constexpr binary_negate< _Predicate > std::not2 (const _Predicate & __pred)`
- `template<typename _Arg, typename _Result>`  
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun ( _Result(* __x)( _Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result>`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun ( _Result(* __x)( _Arg1, _Arg2))`

## Variables

- `template<template< typename > class>`  
`constexpr bool std::__is_std_op_template`
- `template<> constexpr bool std::__is_std_op_template< std::bit_and >`
- `template<> constexpr bool std::__is_std_op_template< std::bit_not >`
- `template<> constexpr bool std::__is_std_op_template< std::bit_or >`
- `template<> constexpr bool std::__is_std_op_template< std::bit_xor >`
- `template<> constexpr bool std::__is_std_op_template< std::divides >`
- `template<> constexpr bool std::__is_std_op_template< std::equal_to >`
- `template<> constexpr bool std::__is_std_op_template< std::greater >`
- `template<> constexpr bool std::__is_std_op_template< std::greater_equal >`
- `template<> constexpr bool std::__is_std_op_template< std::less >`
- `template<> constexpr bool std::__is_std_op_template< std::less_equal >`
- `template<> constexpr bool std::__is_std_op_template< std::logical_and >`
- `template<> constexpr bool std::__is_std_op_template< std::logical_not >`
- `template<> constexpr bool std::__is_std_op_template< std::logical_or >`
- `template<> constexpr bool std::__is_std_op_template< std::minus >`
- `template<> constexpr bool std::__is_std_op_template< std::modulus >`
- `template<> constexpr bool std::__is_std_op_template< std::multiplies >`
- `template<> constexpr bool std::__is_std_op_template< std::negate >`
- `template<> constexpr bool std::__is_std_op_template< std::not_equal_to >`
- `template<> constexpr bool std::__is_std_op_template< std::plus >`
- `template<typename _Fn>`  
`constexpr bool std::__is_std_op_wrapper`
- `template<template< typename > class _Ft, typename _Tp>`  
`constexpr bool std::__is_std_op_wrapper< _Ft< _Tp > >`

### 7.167.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 7.168 `std_heap.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
`constexpr void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance>`  
`constexpr bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`constexpr bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare>`  
`constexpr _Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::__make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>`  
`constexpr void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::__sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator>`  
`constexpr void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵  
comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::push\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵  
comp)`
- `template<typename _RandomAccessIterator>`  
`constexpr void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::sort\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↵  
comp)`

### 7.168.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 7.169 bits/stl\_iterator.h File Reference

### Classes

- class [std::back\\_insert\\_iterator](#)< \_Container >
- class [std::common\\_iterator](#)< \_It, \_Sent >
- class [std::counted\\_iterator](#)< \_It >
- class [std::front\\_insert\\_iterator](#)< \_Container >
- class [std::insert\\_iterator](#)< \_Container >
- class [std::move\\_iterator](#)< \_Iterator >
- class [std::move\\_sentinel](#)< \_Sent >
- class [std::reverse\\_iterator](#)< \_Iterator >

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_MAKE\_MOVE\_IF\_NOEXCEPT\_ITERATOR(_Iter)`
- `#define \_GLIBCXX\_MAKE\_MOVE\_ITERATOR(_Iter)`

### Typedefs

- `template<typename _InputIterator>`  
`using std::\_\_iter\_key\_t`
- `template<typename _InputIterator>`  
`using std::\_\_iter\_to\_alloc\_t`
- `template<typename _InputIterator>`  
`using std::\_\_iter\_val\_t`

- `template<input_iterator _It>`  
    using **std::const\_iterator**
- `template<semiregular _Sent>`  
    using **std::const\_sentinel**
- `template<indirectly_readable _It>`  
    using **std::iter\_const\_reference\_t**

## Functions

- `template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>>`  
    constexpr **std::make\_move\_if\_noexcept\_iterator** (\_Iterator \_\_i)
- `template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>>`  
    constexpr **std::make\_move\_if\_noexcept\_iterator** (\_Tp \*\_\_i)
- `template<typename _Iterator>`  
    constexpr `reverse_iterator<_Iterator>` **std::make\_reverse\_iterator** (\_Iterator \_\_i)
- `template<typename _Container>`  
    constexpr `back_insert_iterator<_Container>` **std::back\_inserter** (\_Container &\_\_x)
- `template<typename _Container>`  
    constexpr `front_insert_iterator<_Container>` **std::front\_inserter** (\_Container &\_\_x)
- `template<typename _Container>`  
    constexpr `insert_iterator<_Container>` **std::inserter** (\_Container &\_\_x, `std::detail::__range_iter_t<_Container>` \_\_i)
- `template<input_iterator _It>`  
    constexpr const\_iterator<\_It> **std::make\_const\_iterator** (\_It \_\_it) noexcept(is\_nothrow\_convertible\_v<\_It, const\_iterator<\_It>>)
- `template<semiregular _Sent>`  
    constexpr const\_sentinel<\_Sent> **std::make\_const\_sentinel** (\_Sent \_\_s) noexcept(is\_nothrow\_convertible\_v<\_Sent, const\_sentinel<\_Sent>>)
- `template<typename _Iterator>`  
    constexpr `move_iterator<_Iterator>` **std::make\_move\_iterator** (\_Iterator \_\_i)
- `template<typename _Iterator>`  
    constexpr `reverse_iterator<_Iterator>` **std::make\_reverse\_iterator** (\_Iterator \_\_i)
- `template<typename _IteratorL, typename _IteratorR>`  
    requires requires { { \_\_x.base() != \_\_y.base() } -> convertible\_to<bool>; }  
    constexpr bool **std::operator!=** (const `reverse_iterator<_IteratorL>` &\_\_x, const `reverse_iterator<_IteratorR>` &\_\_y)
- `template<typename _Iterator>`  
    requires requires { { \_\_x.base() + \_\_n } -> same\_as<\_Iterator>; }  
    constexpr `move_iterator<_Iterator>` **std::operator+** (typename `move_iterator<_Iterator>`::difference\_type \_\_n, const `move_iterator<_Iterator>` &\_\_x)
- `template<typename _Iterator>`  
    constexpr `reverse_iterator<_Iterator>` **std::operator+** (typename `reverse_iterator<_Iterator>`::difference\_type \_\_n, const `reverse_iterator<_Iterator>` &\_\_x)
- `template<typename _IteratorL, typename _IteratorR>`  
    constexpr auto **std::operator-** (const `move_iterator<_IteratorL>` &\_\_x, const `move_iterator<_IteratorR>` &\_\_y) -> decltype(\_\_x.base() - \_\_y.base())
- `template<typename _IteratorL, typename _IteratorR>`  
    constexpr auto **std::operator-** (const `reverse_iterator<_IteratorL>` &\_\_x, const `reverse_iterator<_IteratorR>` &\_\_y) -> decltype(\_\_y.base() - \_\_x.base())

- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator< (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator<= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator<= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator > std::operator<=> (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<=> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<three_way_comparable _Iterator>`  
`constexpr compare\_three\_way\_result\_t< _Iterator, _Iterator > std::operator<=> (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>`  
`constexpr compare\_three\_way\_result\_t< _IteratorL, _IteratorR > std::operator<=> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`constexpr bool std::operator== (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator== (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Iterator>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator== (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator== (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator>= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`

- `template<typename _IteratorL, typename _IteratorR>`  
`requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }`  
`constexpr bool std::operator>= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR`  
`> &__y)`

## Variables

- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool std::disable_sized_sentinel_for< move\_iterator< _Iterator1 >, move\_iterator< _Iterator2`  
`> >`
- `template<typename _Iterator1, typename _Iterator2>`  
`constexpr bool std::disable_sized_sentinel_for< reverse\_iterator< _Iterator1 >, reverse\_iterator< _Iterator2`  
`> >`

### 7.169.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

## 7.170 `debug/stl_iterator.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _Iterator>`  
`constexpr auto __gnu_debug::__base (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_↵`  
`move\_iterator(__base(__it.base())))`
- `template<typename _Iterator, typename _Sequence>`  
`constexpr std::reverse\_iterator< _Iterator > __gnu_debug::__base (const std::reverse\_iterator< \_Safe\_iterator<`  
`_Iterator, _Sequence, std::random\_access\_iterator\_tag > > &__it)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (const std::move\_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const std::move\_iterator< _Iterator > &__it, const std::pair<`  
`_Diff, \_Distance\_precision > &__dist, int __way)`
- `template<typename _Iterator, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (const std::reverse\_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const std::reverse\_iterator< _Iterator > &__it, const std::pair<`  
`_Diff, \_Distance\_precision > &__dist, int __way)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::move\_iterator< ↵`  
`_Iterator > &__first, const std::move\_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::reverse\_iterator<`  
`_Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last)`
- `template<typename _Iterator>`  
`constexpr auto __gnu_debug::__unsafe (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_↵`  
`move\_iterator(__unsafe(__it.base())))`

- `template<typename _Iterator>`  
`constexpr auto __gnu_debug::__unsafe (const std::reverse\_iterator< _Iterator > &__it) -> decltype(std::__↵`  
`make_reverse_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__valid_range (const std::move\_iterator< _Iterator > &__first, const`  
`std::move\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__valid_range (const std::reverse\_iterator< _Iterator > &__first, const`  
`std::reverse\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`

### 7.170.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.171 `std_iterator_base_funcs.h` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp>`  
`ptrdiff_t std::\_\_distance (::List_const_iterator< _Tp >, ::List_const_iterator< _Tp >, input\_iterator\_tag)`
- `template<typename _Tp>`  
`ptrdiff_t std::\_\_distance (::List_iterator< _Tp >, ::List_iterator< _Tp >, input\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::\_\_distance (_InputIterator __first, _InputIterator`  
`__last, input\_iterator\_tag)`
- `template<typename _OutputIterator>`  
`void std::\_\_distance (_OutputIterator, _OutputIterator, output\_iterator\_tag)=delete`
- `template<typename _RandomAccessIterator>`  
`constexpr iterator\_traits< _RandomAccessIterator >::difference_type std::\_\_distance (_RandomAccessIterator`  
`__first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`constexpr void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator>`  
`constexpr iterator\_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __↵`  
`__last)`
- `template<typename _InputIterator>`  
`constexpr _InputIterator std::next (_InputIterator __x, typename iterator\_traits< _InputIterator >::difference_type`  
`__n=1)`
- `template<typename _BidirectionalIterator>`  
`constexpr _BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator\_traits< _Bidirectional↵`  
`Iterator >::difference_type __n=1)`

### 7.171.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

## 7.172 `std_iterator_base_types.h` File Reference

### Classes

- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::contiguous\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::iterator<\\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference>](#)
- struct [std::iterator\\_traits<\\_Iterator>](#)
- struct [std::iterator\\_traits<\\_Tp\\*>](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX26_ALGO_DEF_VAL_T(_Iterator)`
- `#define _GLIBCXX26_DEF_VAL_T(T)`

#### 7.172.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

## 7.173 `std_list.h` File Reference

### Classes

- class [std::\\_List\\_base<\\_Tp, \\_Alloc>](#)
- struct [std::\\_List\\_const\\_iterator<\\_Tp>](#)
- struct [std::\\_List\\_iterator<\\_Tp>](#)
- struct [std::\\_List\\_node<\\_Tp>](#)
- struct [std::\\_\\_detail::\\_List\\_node\\_base](#)
- struct [std::\\_\\_detail::\\_List\\_node\\_header](#)
- class [std::list<\\_Tp, \\_Alloc>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define _GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG`
- `#define _GLIBCXX_USE_ALLOC_PTR_FOR_LIST`



## Functions

- template<typename \_Tp>  
ptrdiff\_t **std::distance** (::List\_const\_iterator< \_Tp >, ::List\_const\_iterator< \_Tp >, input\_iterator\_tag)
- template<typename \_Tp>  
ptrdiff\_t **std::distance** (::List\_iterator< \_Tp >, ::List\_iterator< \_Tp >, input\_iterator\_tag)
- template<bool \_Const, typename \_Ptr>  
ptrdiff\_t **std::distance** (\_\_list::Iterator< \_Const, \_Ptr > \_\_first, \_\_list::Iterator< \_Const, \_Ptr > \_\_last, input\_iterator\_tag \_\_tag)
- template<typename \_InputIterator, typename \_ValT = typename iterator\_traits<\_InputIterator>::value\_type, typename \_Allocator = allocator<\_ValT>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**std::list** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> list<\_ValT, \_Allocator >
- template<typename \_Tp, typename \_Alloc>  
\_\_detail::\_\_synth3way\_t< \_Tp > **std::operator<=>** (const list< \_Tp, \_Alloc > &\_\_x, const list< \_Tp, \_Alloc > &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool **std::operator==** (const list< \_Tp, \_Alloc > &\_\_x, const list< \_Tp, \_Alloc > &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
void **std::swap** (list< \_Tp, \_Alloc > &\_\_x, list< \_Tp, \_Alloc > &\_\_y) noexcept(*/\*conditional \*/*)

### 7.173.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 7.174 std\_map.h File Reference

### Classes

- class **std::map**< \_Key, \_Tp, \_Compare, \_Alloc >

### Namespaces

- namespace **std**

## Functions

- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>>  
**std::map** (\_InputIterator, \_InputIterator, \_Allocator) -> map< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, less< \_\_iter\_key\_t< \_InputIterator > >, \_Allocator >
- template<typename \_InputIterator, typename \_Compare = less<\_\_iter\_key\_t<\_InputIterator>>, typename \_Allocator = allocator<\_\_iter\_val\_t<\_InputIterator>>, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>>  
**std::map** (\_InputIterator, \_InputIterator, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> map< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>>  
**std::map** (initializer\_list< pair< \_Key, \_Tp > >, \_Allocator) -> map< \_Key, \_Tp, less< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Compare = less<\_Key>, typename \_Allocator = allocator<pair<const \_Key, \_Tp>>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>>  
**std::map** (initializer\_list< pair< \_Key, \_Tp > >, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> map< \_Key, \_Tp, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
\_\_detail::\_\_synth3way\_t< pair< const \_Key, \_Tp > > **std::operator<=>** (const map< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_x, const map< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_y)

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`  
`noexcept(/*conditional */)`

### 7.174.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

## 7.175 `std_multimap.h` File Reference

### Classes

- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less< __iter_key_t< _InputIterator > >, typename _Allocator = allocator< __iter_val_t< _InputIterator > >, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::multimap (_InputIterator, _InputIterator, _Compare= _Compare(), _Allocator= _Allocator()) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::multimap (initializer\_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare = less< _Key >, typename _Allocator = allocator< pair< const _Key, _Tp > >, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::multimap (initializer\_list< pair< _Key, _Tp > >, _Compare= _Compare(), _Allocator= _Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 7.175.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

## 7.176 `std_multiset.h` File Reference

### Classes

- class `std::multiset< _Key, _Compare, _Alloc >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator\_traits<_InputIterator>::value_type, less< typename iterator\_traits<_InputIterator>::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator\_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator\_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator\_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::multiset (initializer\_list<_Key >, _Allocator) -> multiset< _Key, less<_Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::multiset (initializer\_list<_Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 7.176.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 7.177 `std_numeric.h` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define \_GLIBCXX\_MOVE\_IF\_20(_E)`

### Functions

- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`  
`constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

### 7.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 7.178 `std_pair.h` File Reference

### Classes

- struct `std::pair<_T1, _T2>`
- struct `std::piecewise_construct_t`
- struct `std::tuple_element<0, pair<_Tp1, _Tp2>>`
- struct `std::tuple_element<1, pair<_Tp1, _Tp2>>`
- struct `std::tuple_size<pair<_Tp1, _Tp2>>`

### Namespaces

- namespace `std`

### Functions

- template<typename \_T1, typename \_T2>  
constexpr `pair<typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type>`  
> `std::make_pair` (\_T1 &&\_\_x, \_T2 &&\_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr const \_Tp && `std::get` (const `pair<_Tp, _Up>` &&\_\_p) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr const \_Tp & `std::get` (const `pair<_Tp, _Up>` &\_\_p) noexcept
- template<size\_t \_Int, class \_Tp1, class \_Tp2>  
constexpr const `tuple_element<_Int, pair<_Tp1, _Tp2>>::type` && `std::get` (const `pair<_Tp1, _Tp2>` &&\_\_in) noexcept
- template<size\_t \_Int, class \_Tp1, class \_Tp2>  
constexpr const `tuple_element<_Int, pair<_Tp1, _Tp2>>::type` & `std::get` (const `pair<_Tp1, _Tp2>` &\_\_in) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr const \_Tp && `std::get` (const `pair<_Up, _Tp>` &&\_\_p) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr const \_Tp & `std::get` (const `pair<_Up, _Tp>` &\_\_p) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr \_Tp && `std::get` (`pair<_Tp, _Up>` &&\_\_p) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr \_Tp & `std::get` (`pair<_Tp, _Up>` &\_\_p) noexcept
- template<size\_t \_Int, class \_Tp1, class \_Tp2>  
constexpr `tuple_element<_Int, pair<_Tp1, _Tp2>>::type` && `std::get` (`pair<_Tp1, _Tp2>` &&\_\_in) noexcept
- template<size\_t \_Int, class \_Tp1, class \_Tp2>  
constexpr `tuple_element<_Int, pair<_Tp1, _Tp2>>::type` & `std::get` (`pair<_Tp1, _Tp2>` &\_\_in) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr \_Tp && `std::get` (`pair<_Up, _Tp>` &&\_\_p) noexcept
- template<typename \_Tp, typename \_Up>  
constexpr \_Tp & `std::get` (`pair<_Up, _Tp>` &\_\_p) noexcept

### Variables

- template<typename \_Tp>  
constexpr bool `std::__is_pair`
- template<typename \_Tp, typename \_Up>  
constexpr bool `std::__is_pair<pair<_Tp, _Up>>`
- constexpr `piecewise_construct_t` `std::piecewise_construct`

- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t std::tuple_size_v< const pair< _Tp1, _Tp2 > >`
- `template<typename _Tp1, typename _Tp2>`  
`constexpr size_t std::tuple_size_v< pair< _Tp1, _Tp2 > >`

### 7.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

## 7.179 `std::queue.h` File Reference

### Classes

- class `std::priority_queue< _Tp, _Sequence, _Compare >`
- class `std::queue< _Tp, _Sequence >`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > std::operator<=> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _Compare >`
- `template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _Container, _Compare >`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>>`  
`std::priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_queue< _ValT, _Container, _Compare >`
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>>`  
`std::queue (_Container) -> queue< typename _Container::value_type, _Container >`

- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>`  
`std::queue` (`_Container`, `_Allocator`) -> `queue< typename _Container::value_type, _Container >`
- `template<typename _Tp, typename _Sequence, typename _Compare>`  
`constexpr enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type`  
`std::swap` (`priority_queue< _Tp, _Sequence, _Compare > &__x`, `priority_queue< _Tp, _Sequence, _Compare > &__y`) `noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type` `std::swap` (`queue< _Tp, _Seq > &__x`, `queue< _Tp, _Seq > &__y`) `noexcept(noexcept(__x.swap(__y)))`

### 7.179.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 7.180 `std_raw_storage_iter.h` File Reference

### Classes

- class `std::raw_storage_iterator< _OutputIterator, _Tp >`

### Namespaces

- namespace `std`

### 7.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.181 `std_relops.h` File Reference

### Namespaces

- namespace `std`
- namespace `std::rel_ops`

### Functions

- `template<class _Tp>`  
`bool std::rel_ops::operator!=` (`const _Tp &__x`, `const _Tp &__y`)
- `template<class _Tp>`  
`bool std::rel_ops::operator<=` (`const _Tp &__x`, `const _Tp &__y`)
- `template<class _Tp>`  
`bool std::rel_ops::operator>` (`const _Tp &__x`, `const _Tp &__y`)
- `template<class _Tp>`  
`bool std::rel_ops::operator>=` (`const _Tp &__x`, `const _Tp &__y`)

### 7.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

This file is only included by `<utility>`, which is required by the standard to define namespace `rel_ops` and its contents.

## 7.182 `std_set.h` File Reference

### Classes

- class `std::set<_Key, _Compare, _Alloc>`

### Namespaces

- namespace `std`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits<_InputIterator>::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>>`  
`std::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

#### 7.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 7.183 `std_stack.h` File Reference

### Classes

- class `std::stack<_Tp, _Sequence>`



## Namespaces

- namespace `std`

## Functions

- `template<typename _Tp, typename _Seq>`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, three_way_comparable _Seq>`  
`compare_three_way_result_t< _Seq > std::operator<=> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq>`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Container, typename = _RequireNotAllocator<_Container>>`  
`std::stack (_Container) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>`  
`std::stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >`
- `template<typename _Tp, typename _Seq>`  
`enable_if< __is_swappable< _Seq >::value >::type std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

### 7.183.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

## 7.184 `std_tempbuf.h` File Reference

### Classes

- class `std::Temporary_buffer< _ForwardIterator, _Tp >`

### Namespaces

- namespace `std`
- namespace `std::__detail`

### Macros

- `#define _GLIBCXX_OPERATOR_DELETE`
- `#define _GLIBCXX_OPERATOR_NEW`
- `#define _GLIBCXX_SIZED_DEALLOC(T, p, n)`

## Functions

- `template<typename _Tp>`  
`_Tp * std::detail::__get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`void std::detail::__return_temporary_buffer (_Tp * __p, size_t __len)`
- `template<typename _Tp, typename _ForwardIterator>`  
`void std::__uninitialized_construct_buf (_Tp * __first, _Tp * __last, _ForwardIterator __seed)`
- `template<typename _Tp>`  
`pair<_Tp *, ptrdiff_t> std::get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`void std::return_temporary_buffer (_Tp * __p)`

### 7.184.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.185 `stl_tree.h` File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_USE_ALLOC_PTR_FOR_RB_TREE`

### Enumerations

- `enum _Rb_tree_color { _S_red, _S_black }`

## Functions

- `unsigned int std::_Rb_tree_black_count (const _Rb_tree_node_base * __node, const _Rb_tree_node_base * __root) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_decrement (_Rb_tree_node_base * __x) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_increment (_Rb_tree_node_base * __x) throw ()`
- `void std::_Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base * __x, _Rb_tree_node_base * __p, _Rb_tree_node_base & __header) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_rebalance_for_erase (_Rb_tree_node_base * const __z, _Rb_tree_node_base & __header) throw ()`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>`  
`void std::swap (_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc> & __x, _Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc> & __y)`

### 7.185.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

## 7.186 `stl_uninitialized.h` File Reference

### Namespaces

- namespace `std`

## Functions

- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_copy\_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void std::uninitialized\_default\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::uninitialized\_default\_construct\_n (_ForwardIterator __first, _Size __count)`
- `template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr void std::uninitialized\_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>`  
`constexpr _ForwardIterator std::uninitialized\_fill\_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _InputIterator, typename _ForwardIterator>`  
`constexpr _ForwardIterator std::uninitialized\_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator>`  
`constexpr pair< _InputIterator, _ForwardIterator > std::uninitialized\_move\_n (_InputIterator __first, _Size __count, _ForwardIterator __result)`
- `template<typename _ForwardIterator>`  
`constexpr void std::uninitialized\_value\_construct (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Size>`  
`constexpr _ForwardIterator std::uninitialized\_value\_construct\_n (_ForwardIterator __first, _Size __count)`

### 7.186.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.187 `std_vector.h` File Reference

### Classes

- struct [std::Vector\\_base](#)< \_Tp, \_Alloc >
- class [std::vector](#)< \_Tp, \_Alloc >

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define GLIBCXX\_ASAN\_ANNOTATE\_BEFORE\_DEALLOC`
- `#define GLIBCXX\_ASAN\_ANNOTATE\_GREW(n)`
- `#define GLIBCXX\_ASAN\_ANNOTATE\_GROW(n)`
- `#define GLIBCXX\_ASAN\_ANNOTATE\_REINIT`
- `#define GLIBCXX\_ASAN\_ANNOTATE\_SHRINK(n)`

## Functions

- `template<typename _Tp, typename _Alloc>  
constexpr __detail::__synth3way_t< _Tp > std::operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>  
constexpr bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>  
constexpr void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>  
std::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >`

### 7.187.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 7.188 `stream_iterator.h` File Reference

### Classes

- class [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >](#)
- class [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)

### 7.188.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.189 `streambuf.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits>  
streamsize std::copy\_streambufs (basic\_streambuf< _CharT, _Traits > *__sbin, basic\_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits>  
streamsize std::copy\_streambufs\_eof (basic\_streambuf< _CharT, _Traits > *__sbin, basic\_streambuf< _CharT, _Traits > *__sbout, bool &__ineof)`

### 7.189.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

## 7.190 streambuf\_iterator.h File Reference

### Classes

- class [std::istreambuf\\_iterator<\\_CharT, \\_Traits>](#)
- class [std::ostreambuf\\_iterator<\\_CharT, \\_Traits>](#)

### Namespaces

- namespace [std](#)

### Functions

- [template<bool \\_IsMove, typename \\_CharT>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, ostreambuf\\_iterator<\\_CharT>>::\\_\\_type std::\\_\\_↵](#)  
**copy\_move\_a2** ([\\_CharT \\* \\_\\_first](#), [\\_CharT \\* \\_\\_last](#), [ostreambuf\\_iterator<\\_CharT> \\_\\_result](#))
- [template<bool \\_IsMove, typename \\_CharT>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, ostreambuf\\_iterator<\\_CharT>>::\\_\\_type std::\\_\\_↵](#)  
**copy\_move\_a2** (const [\\_CharT \\* \\_\\_first](#), const [\\_CharT \\* \\_\\_last](#), [ostreambuf\\_iterator<\\_CharT> \\_\\_result](#))
- [template<bool \\_IsMove, typename \\_CharT>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, \\_CharT \\* >::\\_\\_type std::\\_\\_copy\\_move\\_a2](#)  
([istreambuf\\_iterator<\\_CharT> \\_\\_first](#), [istreambuf\\_iterator<\\_CharT> \\_\\_last](#), [\\_CharT \\* \\_\\_result](#))
- [template<typename \\_CharT, typename \\_Size>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, \\_CharT \\* >::\\_\\_type std::\\_\\_copy\\_n\\_a](#) ([istreambuf\\_iterator<](#)  
[\\_CharT> \\_\\_it](#), [\\_Size \\_\\_n](#), [\\_CharT \\* \\_\\_result](#), bool \_\_strict)
- [template<typename \\_CharT, typename \\_Distance>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, void >::\\_\\_type std::advance](#) ([istreambuf\\_iterator<](#)  
[\\_CharT> &\\_\\_i](#), [\\_Distance \\_\\_n](#))
- [template<typename \\_CharT>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, ostreambuf\\_iterator<\\_CharT>>::\\_\\_type std::copy](#)  
([istreambuf\\_iterator<\\_CharT> \\_\\_first](#), [istreambuf\\_iterator<\\_CharT> \\_\\_last](#), [ostreambuf\\_iterator<\\_CharT>](#)  
[\\_\\_result](#))
- [template<typename \\_CharT>](#)  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if<\\_\\_is\\_char<\\_CharT>::\\_\\_value, istreambuf\\_iterator<\\_CharT>>::\\_\\_type std::find](#)  
([istreambuf\\_iterator<\\_CharT> \\_\\_first](#), [istreambuf\\_iterator<\\_CharT> \\_\\_last](#), const [\\_CharT & \\_\\_val](#))
- [template<typename \\_CharT, typename \\_Traits>](#)  
bool **std::operator==** (const [istreambuf\\_iterator<\\_CharT, \\_Traits> & \\_\\_a](#), const [istreambuf\\_iterator<\\_CharT,](#)  
[\\_Traits> & \\_\\_b](#))

#### 7.190.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 7.191 bits/string\_view.tcc File Reference

### Namespaces

- namespace [std](#)

#### 7.191.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string_view>`.

## 7.192 experimental/bits/string\_view.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

#### 7.192.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/string_view>`.

## 7.193 stringfwd.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef [basic\\_string](#)< char > [std::string](#)
- typedef [basic\\_string](#)< char16\_t > [std::u16string](#)
- typedef [basic\\_string](#)< char32\_t > [std::u32string](#)
- typedef [basic\\_string](#)< wchar\_t > [std::wstring](#)

#### 7.193.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 7.194 text\_encoding-data.h File Reference

### Macros

- `#define _GLIBCXX_TEXT_ENCODING_UTF8_OFFSET`

#### 7.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<text_encoding>`.

## 7.195 this\_thread\_sleep.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::this\\_thread](#)

### Functions

- template<typename \_Rep, typename \_Period>  
void [std::this\\_thread::sleep\\_for](#) (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration>  
void [std::this\\_thread::sleep\\_until](#) (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)

### 7.195.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<thread>`.

## 7.196 unicode-data.h File Reference

### Enumerations

- enum class `_Gcb_property` {  
`_Gcb_Other`, `_Gcb_Control`, `_Gcb_LF`, `_Gcb_CR`,  
`_Gcb_Extend`, `_Gcb_Prepending`, `_Gcb_SpacingMark`, `_Gcb_L`,  
`_Gcb_V`, `_Gcb_T`, `_Gcb_ZWJ`, `_Gcb_LV`,  
`_Gcb_LVT`, `_Gcb_Regional_Indicator` }
- enum class `_InCB` { `_Consonant`, `_Extend` }

### Variables

- constexpr uint32\_t `__escape_edges` []
- constexpr uint32\_t `__gcb_edges` []
- constexpr int `__gcb_shift_bits`
- constexpr uint32\_t `__incb_edges` []
- constexpr char32\_t `__incb_linkers` []
- constexpr char32\_t `__width_edges` []
- constexpr char32\_t `__xpicto_edges` []

### 7.196.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.

## 7.197 unicode.h File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_GET_UNICODE_DATA`

### Typedefs

- template<typename `_View`>  
using `std::__unicode::Utf16_view`
- template<typename `_View`>  
using `std::__unicode::Utf32_view`
- template<typename `_View`>  
using `std::__unicode::Utf8_view`

## Enumerations

- enum class `_Gcb_property` {  
`_Gcb_Other` , `_Gcb_Control` , `_Gcb_LF` , `_Gcb_CR` ,  
`_Gcb_Extend` , `_Gcb_Prepend` , `_Gcb_SpacingMark` , `_Gcb_L` ,  
`_Gcb_V` , `_Gcb_T` , `_Gcb_ZWJ` , `_Gcb_LV` ,  
`_Gcb_LVT` , `_Gcb_Regional_Indicator` }
- enum class `_InCB` { `_Consonant` , `_Extend` }

## Functions

- constexpr bool `std::__unicode::__charset_alias_match` (string\_view \_\_a, string\_view \_\_b)
- template<typename \_CharT>  
constexpr size\_t `std::__unicode::__field_width` (basic\_string\_view< \_CharT > \_\_s)
- constexpr int `std::__unicode::__field_width` (char32\_t \_\_c) noexcept
- constexpr \_Gcb\_property `std::__unicode::__grapheme_cluster_break_property` (char32\_t \_\_c) noexcept
- constexpr \_InCB `std::__unicode::__incb_property` (char32\_t \_\_c) noexcept
- constexpr bool `std::__unicode::__is_extended_pictographic` (char32\_t \_\_c)
- constexpr bool `std::__unicode::__is_incb_linker` (char32\_t \_\_c) noexcept
- constexpr bool `std::__unicode::__is_scalar_value` (char32\_t \_\_c)
- template<typename \_CharT>  
constexpr bool `std::__unicode::__is_single_code_unit` (char32\_t \_\_c)
- constexpr bool `std::__unicode::__literal_encoding_is_extended_ascii` ()
- template<typename \_CharT>  
constexpr bool `std::__unicode::__literal_encoding_is_unicode` ()
- constexpr bool `std::__unicode::__literal_encoding_is_utf8` ()
- constexpr bool `std::__unicode::__should_escape_category` (char32\_t \_\_c) noexcept
- template<typename \_CharT>  
constexpr size\_t `std::__unicode::__truncate` (basic\_string\_view< \_CharT > &\_\_s, size\_t \_\_max)

## Variables

- constexpr uint32\_t `std::__unicode::__escape_edges` []
- constexpr uint32\_t `std::__unicode::__gcb_edges` []
- constexpr int `std::__unicode::__gcb_shift_bits`
- constexpr uint32\_t `std::__unicode::__incb_edges` []
- constexpr char32\_t `std::__unicode::__incb_linkers` []
- constexpr char32\_t `std::__unicode::__width_edges` []
- constexpr char32\_t `std::__unicode::__xpicto_edges` []
- template<typename \_Range>  
constexpr bool `std::ranges::enable_borrowed_range`< `std::__unicode::__Grapheme_cluster_view`< \_↵  
**Range** > >
- template<typename \_To, typename \_Range>  
constexpr bool `std::ranges::enable_borrowed_range`< `std::__unicode::__Utf_view`< \_To, \_Range > >

### 7.197.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<format>`.



## 7.198 uniform\_int\_dist.h File Reference

### Classes

- struct [std::uniform\\_int\\_distribution<\\_IntType>::param\\_type](#)
- class [std::uniform\\_int\\_distribution<\\_IntType>](#)

### Namespaces

- namespace [std](#)

### Concepts

- concept [std::uniform\\_random\\_bit\\_generator](#)

#### 7.198.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 7.199 unique\_lock.h File Reference

### Classes

- class [std::unique\\_lock<\\_Mutex>](#)

### Namespaces

- namespace [std](#)

#### 7.199.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 7.200 unique\_ptr.h File Reference

### Classes

- struct [std::default\\_delete<\\_Tp>](#)
- struct [std::default\\_delete<\\_Tp\[\]>](#)
- struct [std::hash<unique\\_ptr<\\_Tp, \\_Dp>>](#)
- class [std::unique\\_ptr<\\_Tp, \\_Dp>](#)
- class [std::unique\\_ptr<\\_Tp\[\], \\_Dp>](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep>  
constexpr bool [std::operator<](#) (const [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_x, const [unique\\_ptr](#)<\_Up, \_Ep> &\_\_y)
- template<typename \_Tp, typename \_Dp>  
constexpr bool [std::operator<](#) (const [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_x, nullptr\_t)

- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator< (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator<= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator<= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr compare\_three\_way\_result\_t< typename unique\_ptr< _Tp, _Dp >::pointer, typename unique\_ptr< _Up, _Ep >::pointer > std::operator<=> (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr compare\_three\_way\_result\_t< typename unique\_ptr< _Tp, _Dp >::pointer > std::operator<=> (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator== (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator> (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep>`  
`constexpr bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, const unique\_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp>`  
`constexpr bool std::operator>= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp>`  
`bool std::operator>= (nullptr_t, const unique\_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp>`  
`enable\_if<!__is_swappable< _Dp >::value >::type std::swap (unique\_ptr< _Tp, _Dp > &, unique\_ptr< _Tp, _Dp > &)=delete`

### 7.200.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.201 unordered\_map.h File Reference

### Classes

- class [std::unordered\\_map](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multimap](#)< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >

### Namespaces

- namespace [std](#)

## Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__umap_hashtable`
- `template<bool _Cache>  
using std::__umap_traits`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__ummap_hashtable`
- `template<bool _Cache>  
using std::__ummap_traits`

## Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>  
bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>  
bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_↵  
_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>  
void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _↵  
_Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>  
void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _↵  
_Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require_↵  
Allocator<_Allocator>>>  
std::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _Input_↵  
Iterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _↵  
_InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require_↵  
Allocator<_Allocator>>>  
std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator)  
-> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _↵  
_InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _↵  
_RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>  
std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _↵  
Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash,  
equal_to< __iter_key_t< _InputIterator > >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t< _InputIterator>>, typename _Pred = equal_to<__iter_key_↵  
t< _InputIterator>>>, typename _Allocator = allocator<__iter_to_alloc_t< _InputIterator>>, typename = _RequireInputIter<_InputIterator>,  
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↵  
Allocator>>>  
std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _↵  
Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >,  
__iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>  
std::unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _↵  
_Key >, equal_to< _Key >, _Allocator >`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>  
std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type,  
_Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >`

- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_map`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `typename unordered_map`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_map`< `_Key`, `_Tp`, `_Hash`, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_map`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `typename unordered_map`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_map`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multimap`< `__iter_key_t`< `_InputIterator` >, `__iter_val_t`< `_InputIterator` >, `hash`< `__iter_key_t`< `_InputIterator` > >, `equal_to`< `__iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_multimap`< `__iter_key_t`< `_InputIterator` >, `__iter_val_t`< `_InputIterator` >, `hash`< `__iter_key_t`< `_InputIterator` > >, `equal_to`< `__iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap`< `__iter_key_t`< `_InputIterator` >, `__iter_val_t`< `_InputIterator` >, `_Hash`, `equal_to`< `__iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multimap`< `__iter_key_t`< `_InputIterator` >, `__iter_val_t`< `_InputIterator` >, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**`std::unordered_multimap`** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Allocator` >

### 7.201.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

## 7.202 unordered\_set.h File Reference

### Classes

- class `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
- class `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`

### Namespaces

- namespace `std`

### Typedefs

- template<typename \_Value, typename \_Hash = hash<\_Value>, typename \_Pred = std::equal\_to<\_Value>, typename \_Alloc = std::allocator<\_Value>, typename \_Tr = \_\_umset\_traits<\_\_cache\_default<\_Value, \_Hash>::value>>  
using **std::\_\_umset\_hashtable**
- template<bool \_Cache>  
using **std::\_\_umset\_traits**
- template<typename \_Value, typename \_Hash = hash<\_Value>, typename \_Pred = std::equal\_to<\_Value>, typename \_Alloc = std::allocator<\_Value>, typename \_Tr = \_\_uset\_traits<\_\_cache\_default<\_Value, \_Hash>::value>>  
using **std::\_\_uset\_hashtable**
- template<bool \_Cache>  
using **std::\_\_uset\_traits**

### Functions

- template<class \_Value, class \_Hash, class \_Pred, class \_Alloc>  
bool **std::operator==** (const unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, const unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y)
- template<class \_Value, class \_Hash, class \_Pred, class \_Alloc>  
bool **std::operator==** (const unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, const unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y)
- template<class \_Value, class \_Hash, class \_Pred, class \_Alloc>  
void **std::swap** (unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y) noexcept(noexcept(\_\_x.swap(\_\_y)))
- template<class \_Value, class \_Hash, class \_Pred, class \_Alloc>  
void **std::swap** (unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_x, unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > &\_\_y) noexcept(noexcept(\_\_x.swap(\_\_y)))
- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**std::unordered\_multiset** (\_InputIterator, \_InputIterator, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**std::unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, hash< typename iterator\_traits< \_InputIterator >::value\_type >, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >
- template<typename \_InputIterator, typename \_Hash, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocatorOrIntegral<\_Hash>, typename = \_RequireAllocator<\_Allocator>>  
**std::unordered\_multiset** (\_InputIterator, \_InputIterator, unordered\_multiset< int >::size\_type, \_Hash, \_Allocator) -> unordered\_multiset< typename iterator\_traits< \_InputIterator >::value\_type, \_Hash, equal\_to< typename iterator\_traits< \_InputIterator >::value\_type >, \_Allocator >

- [illegible]

- `template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Tp> _Allocator>>`  
`std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >`

### 7.202.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

## 7.203 uses\_allocator\_args.h File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp std::make_obj_using_allocator (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr _Tp * std::uninitialized_construct_using_allocator (_Tp *__p, const _Alloc &__a, _Args &&... __args)`
- `template<_Std_pair _Tp, typename _Alloc>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, _Up &&, _Vp &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, const pair< _Up, _Vp > &) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &&) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &, pair< _Up, _Vp > &) noexcept`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`requires (! _Std_pair<_Tp>)`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, _Args &&... __args) noexcept`
- `template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>`  
`constexpr auto std::uses_allocator_construction_args (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept`

### 7.203.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 7.204 utility.h File Reference

### Classes

- struct `std::integer_sequence< _Tp, _Idx >`

## Namespaces

- namespace [std](#)

## Typedefs

- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::value>::type, size_t = tuple_size<_Tp>::value>`  
`using std::__enable_if_has_tuple_size`
- `template<size_t __i, typename _Tp>`  
`using std::__tuple_element_t`
- `template<size_t... _Idx>`  
`using std::index\_sequence`
- `template<typename... _Types>`  
`using std::index\_sequence\_for`
- `template<size_t _Num>`  
`using std::make\_index\_sequence`
- `template<typename _Tp, _Tp _Num>`  
`using std::make\_integer\_sequence`
- `template<size_t __i, typename _Tp>`  
`using std::tuple_element_t`

## Functions

- `template<typename _Tp, typename... _Types>`  
`constexpr size_t std::__find_uniq_type_in_pack ()`

## Variables

- `template<typename>`  
`constexpr bool std::__is_constant_wrapper_v`
- `template<auto __cw, typename _Fn>`  
`constexpr bool std::__is_constant_wrapper_v< constant\_wrapper< __cw, _Fn > >`
- `template<typename>`  
`constexpr bool std::__is_in_place_index_v`
- `template<size_t _Nm>`  
`constexpr bool std::__is_in_place_index_v< in\_place\_index\_t< _Nm > >`
- `template<typename>`  
`constexpr bool std::__is_in_place_type_v`
- `template<typename _Tp>`  
`constexpr bool std::__is_in_place_type_v< in\_place\_type\_t< _Tp > >`
- `template<typename _Range>`  
`constexpr bool std::ranges::__detail::__is_subrange`
- `template<auto _Xv>`  
`constexpr auto std::cw`
- `constexpr _Swallow_assign std::ignore`
- `constexpr in_place_t std::in_place`
- `template<size_t _Idx>`  
`constexpr in_place_index_t< _Idx > std::in_place_index`
- `template<typename _Tp>`  
`constexpr in_place_type_t< _Tp > std::in_place_type`
- `constexpr sorted_equivalent_t std::sorted_equivalent`
- `constexpr sorted_unique_t std::sorted_unique`
- `template<typename _Tp>`  
`constexpr size_t std::tuple_size_v`



### 7.204.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

This file contains the parts of `<utility>` needed by other headers, so they don't need to include the whole of `<utility>`.

## 7.205 valarray\_after.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

### Functions

- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ↵`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↵`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::↵`  
`atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename ↵`  
`_Dom2::value_type > &__e2)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`_Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename ↵`  
`_Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray<`  
`_Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`_Dom::value_type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const ↵`  
`Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const`  
`_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↵`  
`__not_equal_to, typename _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, type-`  
`name _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__modulus, typename _Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__bitwise_and, typename _Dom1::value_type >::result_type > std::operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`logical_and, typename _Dom1::value_type >::result_type > std::operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`multiplies, typename _Dom1::value_type >::result_type > std::operator* (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __`  
`fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __`  
`fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus,`  
`typename _Dom1::value_type >::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::`  
`value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __`  
`fun< struct std::__plus, typename _Dom::value_type >::result_type > std::operator+ (const typename __`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type>::result_type> std::operator+ (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator- (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator- (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type>::result_type> std::operator- (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::result_type> std::operator- (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::result_type> std::operator- (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator/ (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator/ (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2>, typename _Dom1::value_type>::result_type> std::operator/ (const _Expr< _Dom1, typename _Dom1::value_type> &__v, const _Expr< _Dom2, typename _Dom2::value_type> &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::result_type> std::operator/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom>, typename _Dom::value_type>::result_type> std::operator/ (const valarray< typename _Dom::value_type> &__v, const _Expr< _Dom, typename _Dom::value_type> &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator< (const _Expr< _Dom, typename _Dom::value_type> &__e, const valarray< typename _Dom::value_type> &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type>, typename _Dom::value_type>::result_type> std::operator< (const _Expr< _Dom, typename _Dom::value_type> &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less,`  
`typename _Dom1::value_type >::result_type > std::operator< (const _Expr< _Dom1, typename _Dom1::↵`  
`value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const typename ↵`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ↵`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↵`  
`__shift_left, typename _Dom1::value_type >::result_type > std::operator<< (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↵`  
`less_equal, typename _Dom1::value_type >::result_type > std::operator<= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`



- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`_equal_to, typename _Dom1::value_type >::result_type > std::operator== (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _`  
`Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`greater, typename _Dom1::value_type >::result_type > std::operator> (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >,`  
`typename _fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`greater_equal, typename _Dom1::value_type >::result_type > std::operator>= (const _Expr< _Dom1, type-`  
`name _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >,`  
`typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`shift_right, typename _Dom1::value_type >::result_type > std::operator>> (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`bitwise_xor, typename _Dom1::value_type >::result_type > std::operator^ (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`



- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`_bitwise_or, typename _Dom1::value_type >::result_type > std::operator| (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`_logical_or, typename _Dom1::value_type >::result_type > std::operator|| (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type-`  
`name _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2>`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::pow`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::`  
`value_type > &__e2)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::__Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_Dom::value_type > std::pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`

- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp`  
`> &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > std::pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > std::tan (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
- `template<class _Dom>`  
`_Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > std::tanh (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp>`  
`_Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

### 7.205.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.206 valarray\_array.h File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

## Functions

- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp>`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp>`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp>`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`

- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `template<typename _Tp>`  
`_Tp * std::__valarray_get_storage (size_t)`
- `template<typename _Ta>`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta>`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp>`  
`void std::__valarray_release_memory (_Tp *__p, size_t __n)`
- `template<typename _Tp>`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp >`  
`&__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t`  
`__n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t`  
`__n)`

- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`  
`__n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`  
`> __i)`
- `template<typename _Tp>`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp>`  
`void std::Array_augmented___modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___modulus (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented___plus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`



- `template<typename _Tp>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::_Array_augmented___plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::_Array_augmented___shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp>`  
`void std::_Array_augmented___shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::__Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

### 7.206.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.207 valarray\_array.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`

- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp>`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp>`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

### 7.207.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.208 `valarray_before.h` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### 7.208.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 7.209 `vector.tcc` File Reference

### Namespaces

- namespace [std](#)

### 7.209.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 7.210 `version.h` File Reference

### Macros

- `#define __glibcxx_adaptor_iterator_pair_constructor`
- `#define __glibcxx_addressof_constexpr`
- `#define __glibcxx_algorithm_default_value_type`
- `#define __glibcxx_algorithm_iterator_requirements`
- `#define __glibcxx_aligned_accessor`
- `#define __glibcxx_allocate_at_least`
- `#define __glibcxx_allocator_traits_is_always_equal`

- `#define __glibcxx_any`
- `#define __glibcxx_apply`
- `#define __glibcxx_array_constexpr`
- `#define __glibcxx_as_const`
- `#define __glibcxx_associative_heterogeneous_erasure`
- `#define __glibcxx_associative_heterogeneous_insertion`
- `#define __glibcxx_assume_aligned`
- `#define __glibcxx_atomic_flag_test`
- `#define __glibcxx_atomic_float`
- `#define __glibcxx_atomic_is_always_lock_free`
- `#define __glibcxx_atomic_min_max`
- `#define __glibcxx_atomic_ref`
- `#define __glibcxx_atomic_shared_ptr`
- `#define __glibcxx_atomic_value_initialization`
- `#define __glibcxx_atomic_wait`
- `#define __glibcxx_barrier`
- `#define __glibcxx_bind_back`
- `#define __glibcxx_bind_front`
- `#define __glibcxx_bit_cast`
- `#define __glibcxx_bitops`
- `#define __glibcxx_bitset`
- `#define __glibcxx_bool_constant`
- `#define __glibcxx_bounded_array_traits`
- `#define __glibcxx_boyer_moore_searcher`
- `#define __glibcxx_byte`
- `#define __glibcxx_byteswap`
- `#define __glibcxx_chrono`
- `#define __glibcxx_chrono_cxx20`
- `#define __glibcxx_chrono_udls`
- `#define __glibcxx_clamp`
- `#define __glibcxx_common_reference`
- `#define __glibcxx_common_reference_wrapper`
- `#define __glibcxx_complex_udls`
- `#define __glibcxx_concepts`
- `#define __glibcxx_constant_wrapper`
- `#define __glibcxx_constexpr_algorithms`
- `#define __glibcxx_constexpr_bitset`
- `#define __glibcxx_constexpr_char_traits`
- `#define __glibcxx_constexpr_charconv`
- `#define __glibcxx_constexpr_complex`
- `#define __glibcxx_constexpr_dynamic_alloc`
- `#define __glibcxx_constexpr_flat_map`
- `#define __glibcxx_constexpr_flat_set`
- `#define __glibcxx_constexpr_functional`
- `#define __glibcxx_constexpr_iterator`
- `#define __glibcxx_constexpr_memory`
- `#define __glibcxx_constexpr_numeric`
- `#define __glibcxx_constexpr_string`
- `#define __glibcxx_constexpr_string_view`
- `#define __glibcxx_constexpr_tuple`
- `#define __glibcxx_constexpr_typeinfo`

- `#define __glibcxx_constexpr_utility`
- `#define __glibcxx_constexpr_vector`
- `#define __glibcxx_constrained_equality`
- `#define __glibcxx_containers_ranges`
- `#define __glibcxx_copyable_function`
- `#define __glibcxx_coroutine`
- `#define __glibcxx_debugging`
- `#define __glibcxx_enable_shared_from_this`
- `#define __glibcxx_endian`
- `#define __glibcxx_erase_if`
- `#define __glibcxx_exception_ptr_cast`
- `#define __glibcxx_exchange_function`
- `#define __glibcxx_execution`
- `#define __glibcxx_expected`
- `#define __glibcxx_filesystem`
- `#define __glibcxx_flat_map`
- `#define __glibcxx_flat_set`
- `#define __glibcxx_format`
- `#define __glibcxx_format_path`
- `#define __glibcxx_format_ranges`
- `#define __glibcxx_format_uchar`
- `#define __glibcxx_formatters`
- `#define __glibcxx_forward_like`
- `#define __glibcxx_freestanding_algorithm`
- `#define __glibcxx_freestanding_array`
- `#define __glibcxx_freestanding_cstring`
- `#define __glibcxx_freestanding_optional`
- `#define __glibcxx_freestanding_string_view`
- `#define __glibcxx_freestanding_variant`
- `#define __glibcxx_fstream_native_handle`
- `#define __glibcxx_function_ref`
- `#define __glibcxx_gcd`
- `#define __glibcxx_gcd_lcm`
- `#define __glibcxx_generator`
- `#define __glibcxx_generic_associative_lookup`
- `#define __glibcxx_generic_unordered_lookup`
- `#define __glibcxx_has_unique_object_representations`
- `#define __glibcxx_hypot`
- `#define __glibcxx_incomplete_container_elements`
- `#define __glibcxx_indirect`
- `#define __glibcxx_initializer_list`
- `#define __glibcxx_inplace_vector`
- `#define __glibcxx_int_pow2`
- `#define __glibcxx_integer_comparison_functions`
- `#define __glibcxx_integer_sequence`
- `#define __glibcxx_integral_constant_callable`
- `#define __glibcxx_interpolate`
- `#define __glibcxx_invoke`
- `#define __glibcxx_invoke_r`
- `#define __glibcxx_ios_noreplace`
- `#define __glibcxx_is_aggregate`

- #define \_\_glibcxx\_is\_constant\_evaluated
- #define \_\_glibcxx\_is\_final
- #define \_\_glibcxx\_is\_implicit\_lifetime
- #define \_\_glibcxx\_is\_invocable
- #define \_\_glibcxx\_is\_layout\_compatible
- #define \_\_glibcxx\_is\_nothrow\_convertible
- #define \_\_glibcxx\_is\_null\_pointer
- #define \_\_glibcxx\_is\_pointer\_interconvertible
- #define \_\_glibcxx\_is\_scoped\_enum
- #define \_\_glibcxx\_is\_structural
- #define \_\_glibcxx\_is\_sufficiently\_aligned
- #define \_\_glibcxx\_is\_swappable
- #define \_\_glibcxx\_is\_virtual\_base\_of
- #define \_\_glibcxx\_jthread
- #define \_\_glibcxx\_latch
- #define \_\_glibcxx\_launder
- #define \_\_glibcxx\_lcm
- #define \_\_glibcxx\_list\_remove\_return\_type
- #define \_\_glibcxx\_logical\_traits
- #define \_\_glibcxx\_make\_from\_tuple
- #define \_\_glibcxx\_make\_obj\_using\_allocator
- #define \_\_glibcxx\_make\_reverse\_iterator
- #define \_\_glibcxx\_make\_unique
- #define \_\_glibcxx\_map\_try\_emplace
- #define \_\_glibcxx\_math\_constants
- #define \_\_glibcxx\_math\_spec\_funcs
- #define \_\_glibcxx\_math\_special\_functions
- #define \_\_glibcxx\_mdspan
- #define \_\_glibcxx\_memory\_resource
- #define \_\_glibcxx\_move\_iterator\_concept
- #define \_\_glibcxx\_move\_only\_function
- #define \_\_glibcxx\_node\_extract
- #define \_\_glibcxx\_nonmember\_container\_access
- #define \_\_glibcxx\_not\_fn
- #define \_\_glibcxx\_null\_iterators
- #define \_\_glibcxx\_observable\_checkpoint
- #define \_\_glibcxx\_optional
- #define \_\_glibcxx\_optional\_range\_support
- #define \_\_glibcxx\_out\_ptr
- #define \_\_glibcxx\_padded\_layouts
- #define \_\_glibcxx\_parallel\_algorithm
- #define \_\_glibcxx\_philox\_engine
- #define \_\_glibcxx\_polymorphic
- #define \_\_glibcxx\_polymorphic\_allocator
- #define \_\_glibcxx\_print
- #define \_\_glibcxx\_quoted\_string\_io
- #define \_\_glibcxx\_ranges
- #define \_\_glibcxx\_ranges\_as\_const
- #define \_\_glibcxx\_ranges\_as\_input
- #define \_\_glibcxx\_ranges\_as\_rvalue
- #define \_\_glibcxx\_ranges\_cache\_latest

- `#define __glibcxx_ranges_cartesian_product`
- `#define __glibcxx_ranges_chunk`
- `#define __glibcxx_ranges_chunk_by`
- `#define __glibcxx_ranges_contains`
- `#define __glibcxx_ranges_enumerate`
- `#define __glibcxx_ranges_filter`
- `#define __glibcxx_ranges_find_last`
- `#define __glibcxx_ranges_fold`
- `#define __glibcxx_ranges_indices`
- `#define __glibcxx_ranges_iota`
- `#define __glibcxx_ranges_join_with`
- `#define __glibcxx_ranges_repeat`
- `#define __glibcxx_ranges_slide`
- `#define __glibcxx_ranges_starts_ends_with`
- `#define __glibcxx_ranges_stride`
- `#define __glibcxx_ranges_to_container`
- `#define __glibcxx_ranges_zip`
- `#define __glibcxx_ratio`
- `#define __glibcxx_raw_memory_algorithms`
- `#define __glibcxx_reference_from_temporary`
- `#define __glibcxx_reference_wrapper`
- `#define __glibcxx_remove_cvref`
- `#define __glibcxx_result_of_sfinae`
- `#define __glibcxx_robust_nonmodifying_seq_ops`
- `#define __glibcxx_sample`
- `#define __glibcxx_saturation_arithmetic`
- `#define __glibcxx_scoped_lock`
- `#define __glibcxx_semaphore`
- `#define __glibcxx_shared_mutex`
- `#define __glibcxx_shared_ptr_arrays`
- `#define __glibcxx_shared_ptr_weak_type`
- `#define __glibcxx_shared_timed_mutex`
- `#define __glibcxx_shift`
- `#define __glibcxx_smart_ptr_for_overwrite`
- `#define __glibcxx_smart_ptr_owner_equality`
- `#define __glibcxx_source_location`
- `#define __glibcxx_span`
- `#define __glibcxx_spanstream`
- `#define __glibcxx_ssize`
- `#define __glibcxx_sstream_from_string_view`
- `#define __glibcxx_start_lifetime_as`
- `#define __glibcxx_starts_ends_with`
- `#define __glibcxx_stdatomic_h`
- `#define __glibcxx_stdbit_h`
- `#define __glibcxx_stdckdint_h`
- `#define __glibcxx_string_contains`
- `#define __glibcxx_string_resize_and_overwrite`
- `#define __glibcxx_string_subview`
- `#define __glibcxx_string_udls`
- `#define __glibcxx_string_view`
- `#define __glibcxx_submdspan`

- `#define __glibcxx_three_way_comparison`
- `#define __glibcxx_to_address`
- `#define __glibcxx_to_underlying`
- `#define __glibcxx_transformation_trait_aliases`
- `#define __glibcxx_transparent_operators`
- `#define __glibcxx_tuple_element_t`
- `#define __glibcxx_tuple_like`
- `#define __glibcxx_tuples_by_type`
- `#define __glibcxx_type_identity`
- `#define __glibcxx_type_trait_variable_templates`
- `#define __glibcxx_uncaught_exceptions`
- `#define __glibcxx_unordered_map_try_emplace`
- `#define __glibcxx_unreachable`
- `#define __glibcxx_unwrap_ref`
- `#define __glibcxx_valarray`
- `#define __glibcxx_variant`
- `#define __glibcxx_void_t`

### 7.210.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

## 7.211 `bitset` File Reference

### Classes

- struct [std::\\_Base\\_bitset< \\_Nw >](#)
- struct [std::\\_Base\\_bitset< 0 >](#)
- struct [std::\\_Base\\_bitset< 1 >](#)
- class [std::bitset< \\_Nb >](#)
- struct [std::hash<::bitset< \\_Nb > >](#)
- class [std::bitset< \\_Nb >::reference](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_BITSET_BITS_PER_ULL`
- `#define _GLIBCXX_BITSET_BITS_PER_WORD`
- `#define _GLIBCXX_BITSET_WORDS(__n)`

### Typedefs

- `template<typename _CharT>`  
`using std::\_bitset::\_\_string`



## Functions

- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
  
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`

### 7.211.1 Detailed Description

This is a Standard C++ Library header.

## 7.212 debug/bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset< \\_Nb >](#)
- struct [std::hash< \\_\\_debug::bitset< \\_Nb > >](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::\_\_debug::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & std::\_\_debug::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & std::\_\_debug::operator>> (std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::\_\_debug::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`constexpr bitset< _Nb > std::\_\_debug::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`

### 7.212.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.213 `cassert` File Reference

### 7.213.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.214 `ccomplex` File Reference

### Macros

- `#define _GLIBCXX_CCOMPLEX`

### 7.214.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 7.215 `tr1/ccomplex` File Reference

### Macros

- `#define _GLIBCXX_TR1_CCOMPLEX`

### 7.215.1 Detailed Description

This is a TR1 C++ Library header.

## 7.216 `cctype` File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CCTYPE`

### 7.216.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `cctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.217 `tr1/cctype` File Reference

### Macros

- `#define _GLIBCXX_TR1_CCTYPE`

### 7.217.1 Detailed Description

This is a TR1 C++ Library header.

## 7.218 cerrno File Reference

### Macros

- #define `_GLIBCXX_CERRNO`
- #define `errno`

#### 7.218.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.219 cfnv File Reference

### Macros

- #define `_GLIBCXX_CFENV`

#### 7.219.1 Detailed Description

This is a Standard C++ Library header.

## 7.220 tr1/cfnv File Reference

### Macros

- #define `_GLIBCXX_TR1_CFENV`

#### 7.220.1 Detailed Description

This is a TR1 C++ Library header.

## 7.221 cfloat File Reference

### Macros

- #define `_GLIBCXX_CFLOAT`
- #define `DECIMAL_DIG`
- #define `FLT_EVAL_METHOD`

#### 7.221.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.222 tr1/cfloat File Reference

### Macros

- #define `_GLIBCXX_TR1_CFLOAT`

### 7.222.1 Detailed Description

This is a TR1 C++ Library header.

## 7.223 charconv File Reference

### Classes

- struct [std::from\\_chars\\_result](#)
- struct [std::to\\_chars\\_result](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define _GLIBCXX_TO_CHARS(T)`

### Typedefs

- `template<typename _Tp>`  
`using std::\_\_detail::\_\_unsigned\_least\_t`

### Enumerations

- enum class [std::chars\\_format](#) { [scientific](#) , [fixed](#) , [hex](#) , [general](#) }

### Functions

- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool std::\_\_detail::\_\_from\_chars\_alnum (const char *__first, const char *__last, _Tp &__val, int __base)`
- `template<bool _DecOnly = false>`  
`constexpr unsigned char std::\_\_detail::\_\_from\_chars\_alnum\_to\_val (unsigned char __c)`
- `from\_chars\_result std::\_\_from\_chars\_bfloat16\_t (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars\_format::general) noexcept`
- `from\_chars\_result std::\_\_from\_chars\_float16\_t (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars\_format::general) noexcept`
- `template<bool _DecOnly, typename _Tp>`  
`constexpr bool std::\_\_detail::\_\_from\_chars\_pow2\_base (const char *__first, const char *__last, _Tp &__val, int __base)`
- `template<typename _Tp>`  
`constexpr bool std::\_\_detail::\_\_raise\_and\_add (_Tp &__val, int __base, unsigned char __c)`
- `template<typename _Tp>`  
`constexpr to\_chars\_result std::\_\_detail::\_\_to\_chars (char *__first, char *__last, _Tp __val, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr to\_chars\_result std::\_\_detail::\_\_to\_chars\_10 (char *__first, char *__last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to\_chars\_result std::\_\_detail::\_\_to\_chars\_16 (char *__first, char *__last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to\_chars\_result std::\_\_detail::\_\_to\_chars\_2 (char *__first, char *__last, _Tp __val) noexcept`
- `template<typename _Tp>`  
`constexpr to\_chars\_result std::\_\_detail::\_\_to\_chars\_8 (char *__first, char *__last, _Tp __val) noexcept`

- `template<typename _Tp>`  
`constexpr to\_chars\_result std::to_chars_i (char *__first, char *__last, _Tp __value, int __base=10)`
- `template<typename _Tp>`  
`constexpr unsigned std::detail::to_chars_len (_Tp __value, int __base) noexcept`
- `template<typename _Tp>`  
`constexpr unsigned std::detail::to_chars_len_2 (_Tp __value) noexcept`
- `template<typename _Tp, enable_if_t<__or_<__is_signed_or_unsigned_integer<_Tp>, is_same<char, _Tp>>::value, int>=0>`  
`constexpr from\_chars\_result std::from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)`
- `from\_chars\_result std::from_chars (const char *__first, const char *__last, double &__value, chars\_format __fmt=chars_format::general) noexcept`
- `from\_chars\_result std::from_chars (const char *__first, const char *__last, float &__value, chars\_format __fmt=chars_format::general) noexcept`
- `from\_chars\_result std::from_chars (const char *__first, const char *__last, long double &__value, chars\_format __fmt=chars_format::general) noexcept`
- `constexpr chars\_format std::operator& (chars\_format __lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format & std::operator&= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format std::operator^ (chars\_format __lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format & std::operator^= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format std::operator| (chars\_format __lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format & std::operator|= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `constexpr chars\_format std::operator~ (chars\_format __fmt) noexcept`
- `to\_chars\_result std::to_chars (char *, char *, bool, int=10)=delete`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, char __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, signed char __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, signed int __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, signed long __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, signed long long __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, signed short __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, unsigned char __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, unsigned int __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, unsigned long __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, unsigned long long __value, int __base=10)`
- `constexpr to\_chars\_result std::to_chars (char *__first, char *__last, unsigned short __value, int __base=10)`

### 7.223.1 Detailed Description

This is a Standard C++ Library header.

## 7.224 chrono File Reference

### Classes

- class `std::chrono::tzdb_list::const_iterator`
- class `std::chrono::gps_clock`
- class `std::chrono::hh_mm_ss<_Duration>`
- class `std::chrono::tai_clock`
- class `std::chrono::tzdb_list`
- class `std::chrono::utc_clock`

## Namespaces

- namespace [std](#)
- namespace [std::chrono](#)
- namespace [std::literals](#)
- namespace [std::literals::chrono\\_literals](#)

## Typedefs

- using [std::chrono::gps\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [std::chrono::gps\\_time](#)
- using [std::chrono::local\\_days](#)
- using [std::chrono::local\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [std::chrono::local\\_time](#)
- using [std::chrono::tai\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [std::chrono::tai\\_time](#)
- using [std::chrono::utc\\_seconds](#)
- template<typename [\\_Duration](#)>  
using [std::chrono::utc\\_time](#)
- using [std::chrono::zoned\\_seconds](#)

## Enumerations

- enum class [choose](#) { [earliest](#) , [latest](#) }

## Functions

- leap\_second\_info [std::chrono::\\_\\_detail::\\_\\_get\\_leap\\_second\\_info](#) (sys\_seconds [\\_\\_ss](#), bool [\\_\\_is\\_utc](#))
- template<typename [\\_DestClock](#), typename [\\_SourceClock](#), typename [\\_Duration](#)>  
requires [\\_\\_detail::\\_\\_clock\\_convs<\\_DestClock, \\_SourceClock, \\_Duration>](#) || [\\_\\_detail::\\_\\_clock\\_convs\\_sys<\\_DestClock, \\_SourceClock, \\_Duration>](#) || [\\_\\_detail::\\_\\_clock\\_convs\\_utc<\\_DestClock, \\_SourceClock, \\_Duration>](#) || [\\_\\_detail::\\_\\_clock\\_convs\\_sys\\_utc<\\_DestClock, \\_SourceClock, \\_Duration>](#) || [\\_\\_detail::\\_\\_clock\\_convs\\_utc\\_sys<\\_DestClock, \\_SourceClock, \\_Duration>](#)  
auto [std::chrono::clock\\_cast](#) (const [time\\_point<\\_SourceClock, \\_Duration>](#) & [\\_\\_t](#))
- const time\_zone \* [std::chrono::current\\_zone](#) ()
- template<typename [\\_Duration](#)>  
leap\_second\_info [std::chrono::get\\_leap\\_second\\_info](#) (const utc\_time< [\\_Duration](#) > & [\\_\\_ut](#))
- const tzdb & [std::chrono::get\\_tzdb](#) ()
- [tzdb\\_list](#) & [std::chrono::get\\_tzdb\\_list](#) ()
- constexpr bool [std::chrono::is\\_am](#) (const [hours](#) & [\\_\\_h](#)) noexcept
- constexpr bool [std::chrono::is\\_pm](#) (const [hours](#) & [\\_\\_h](#)) noexcept
- const time\_zone \* [std::chrono::locate\\_zone](#) (string\_view [\\_\\_tz\\_name](#))
- constexpr [hours](#) [std::chrono::make12](#) (const [hours](#) & [\\_\\_h](#)) noexcept
- constexpr [hours](#) [std::chrono::make24](#) (const [hours](#) & [\\_\\_h](#), bool [\\_\\_is\\_pm](#)) noexcept
- constexpr chrono::day [std::literals::chrono\\_literals::operator""d](#) (unsigned long long [\\_\\_d](#)) noexcept
- constexpr chrono::year [std::literals::chrono\\_literals::operator""y](#) (unsigned long long [\\_\\_y](#)) noexcept
- template<typename [\\_Dur1](#), typename [\\_TZPtr1](#), typename [\\_Dur2](#), typename [\\_TZPtr2](#)>  
bool [std::chrono::operator==](#) (const zoned\_time< [\\_Dur1](#), [\\_TZPtr1](#) > & [\\_\\_x](#), const zoned\_time< [\\_Dur2](#), [\\_TZPtr2](#) > & [\\_\\_y](#))
- const tzdb & [std::chrono::reload\\_tzdb](#) ()
- [string](#) [std::chrono::remote\\_version](#) ()

- **std::chrono::zoned\_time** () -> zoned\_time< [seconds](#) >
- template<typename \_TimeZonePtrOrName>  
**std::chrono::zoned\_time** (\_TimeZonePtrOrName &&) -> zoned\_time< [seconds](#), \_\_time\_zone\_representation< \_TimeZonePtrOrName > >
- template<typename \_TimeZonePtrOrName, typename \_Duration>  
**std::chrono::zoned\_time** (\_TimeZonePtrOrName &&, local\_time< \_Duration >, choose=choose::earliest) -> zoned\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) >, \_\_time\_zone\_representation< \_TimeZonePtrOrName > >
- template<typename \_TimeZonePtrOrName, typename \_Duration>  
**std::chrono::zoned\_time** (\_TimeZonePtrOrName &&, sys\_time< \_Duration >) -> zoned\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) >, \_\_time\_zone\_representation< \_TimeZonePtrOrName > >
- template<typename \_Duration, typename \_TimeZonePtrOrName, typename \_TimeZonePtr2>  
**std::chrono::zoned\_time** (\_TimeZonePtrOrName &&, zoned\_time< \_Duration, \_TimeZonePtr2 >, choose=choose::earliest) -> zoned\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) >, \_\_time\_zone\_representation< \_TimeZonePtrOrName > >
- template<typename \_Duration>  
**std::chrono::zoned\_time** (sys\_time< \_Duration >) -> zoned\_time< [common\\_type\\_t](#)< \_Duration, [seconds](#) >

## Variables

- constexpr month **std::chrono::April**
- constexpr month **std::chrono::August**
- constexpr month **std::chrono::December**
- constexpr month **std::chrono::February**
- constexpr weekday **std::chrono::Friday**
- template<> constexpr bool **std::chrono::is\_clock\_v**< [gps\\_clock](#) >
- template<> constexpr bool **std::chrono::is\_clock\_v**< [tai\\_clock](#) >
- template<> constexpr bool **std::chrono::is\_clock\_v**< [utc\\_clock](#) >
- constexpr month **std::chrono::January**
- constexpr month **std::chrono::July**
- constexpr month **std::chrono::June**
- constexpr last\_spec **std::chrono::last**
- constexpr month **std::chrono::March**
- constexpr month **std::chrono::May**
- constexpr weekday **std::chrono::Monday**
- constexpr month **std::chrono::November**
- constexpr month **std::chrono::October**
- constexpr weekday **std::chrono::Saturday**
- constexpr month **std::chrono::September**
- constexpr weekday **std::chrono::Sunday**
- constexpr weekday **std::chrono::Thursday**
- constexpr weekday **std::chrono::Tuesday**
- constexpr weekday **std::chrono::Wednesday**

### 7.224.1 Detailed Description

This is a Standard C++ Library header.

## 7.225 experimental/chrono File Reference

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_CHRONO`

### Variables

- `template<typename _Rep>`  
`constexpr bool std::chrono::experimental::treat_as_floating_point_v`

#### 7.225.1 Detailed Description

This is a TS C++ Library header.

### 7.226 cinttypes File Reference

#### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CINTTYPES`

#### 7.226.1 Detailed Description

This is a Standard C++ Library header.

### 7.227 tr1/cinttypes File Reference

#### Macros

- `#define _GLIBCXX_TR1_CINTTYPES`

#### 7.227.1 Detailed Description

This is a TR1 C++ Library header.

### 7.228 iso646 File Reference

#### 7.228.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

Since

C++11 (removed in C++20)

### 7.229 climits File Reference

#### Macros

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`



### 7.229.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.230 tr1/climits File Reference

### Macros

- `#define _GLIBCXX_TR1_CLIMITS`

### 7.230.1 Detailed Description

This is a TR1 C++ Library header.

## 7.231 clocale File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CLOCALE`

### 7.231.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.232 cmath File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CMATH`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

### Functions

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acos (_Tp __x)`
- `constexpr float std::acos (float __x)`
- `constexpr long double std::acos (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acosh (_Tp __x)`
- `constexpr float std::acosh (float __x)`
- `constexpr long double std::acosh (long double __x)`

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asin (_Tp __x)`
- `constexpr float std::asin (float __x)`
- `constexpr long double std::asin (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asinh (_Tp __x)`
- `constexpr float std::asinh (float __x)`
- `constexpr long double std::asinh (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atan (_Tp __x)`
- `constexpr float std::atan (float __x)`
- `constexpr long double std::atan (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::atan2 (_Tp __y, _Up __x)`
- `constexpr float std::atan2 (float __y, float __x)`
- `constexpr long double std::atan2 (long double __y, long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atanh (_Tp __x)`
- `constexpr float std::atanh (float __x)`
- `constexpr long double std::atanh (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cbrt (_Tp __x)`
- `constexpr float std::cbrt (float __x)`
- `constexpr long double std::cbrt (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::ceil (_Tp __x)`
- `constexpr float std::ceil (float __x)`
- `constexpr long double std::ceil (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::copysign (_Tp __x, _Up __y)`
- `constexpr float std::copysign (float __x, float __y)`
- `constexpr long double std::copysign (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cos (_Tp __x)`
- `constexpr float std::cos (float __x)`
- `constexpr long double std::cos (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cosh (_Tp __x)`
- `constexpr float std::cosh (float __x)`
- `constexpr long double std::cosh (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::erf (_Tp __x)`
- `constexpr float std::erf (float __x)`
- `constexpr long double std::erf (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::erfc (_Tp __x)`
- `constexpr float std::erfc (float __x)`
- `constexpr long double std::erfc (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::exp (_Tp __x)`
- `constexpr float std::exp (float __x)`

- constexpr long double **std::exp** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::exp2** (\_Tp \_\_x)
- constexpr float **std::exp2** (float \_\_x)
- constexpr long double **std::exp2** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::expm1** (\_Tp \_\_x)
- constexpr float **std::expm1** (float \_\_x)
- constexpr long double **std::expm1** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::fabs** (\_Tp \_\_x)
- constexpr float **std::fabs** (float \_\_x)
- constexpr long double **std::fabs** (long double \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fdim** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::fdim** (float \_\_x, float \_\_y)
- constexpr long double **std::fdim** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::floor** (\_Tp \_\_x)
- constexpr float **std::floor** (float \_\_x)
- constexpr long double **std::floor** (long double \_\_x)
- template<typename \_Tp, typename \_Up, typename \_Vp>  
constexpr \_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Up, \_Vp >::\_\_type **std::fma** (\_Tp \_\_x, \_Up \_\_y, \_Vp \_\_z)
- constexpr float **std::fma** (float \_\_x, float \_\_y, float \_\_z)
- constexpr long double **std::fma** (long double \_\_x, long double \_\_y, long double \_\_z)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmax** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::fmax** (float \_\_x, float \_\_y)
- constexpr long double **std::fmax** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmin** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::fmin** (float \_\_x, float \_\_y)
- constexpr long double **std::fmin** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmod** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::fmod** (float \_\_x, float \_\_y)
- constexpr long double **std::fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, int >::\_\_type **std::fpclassify** (\_Tp \_\_x)
- constexpr int **std::fpclassify** (double \_\_x)
- constexpr int **std::fpclassify** (float \_\_x)
- constexpr int **std::fpclassify** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::frexp** (\_Tp \_\_x, int \* \_\_exp)
- float **std::frexp** (float \_\_x, int \* \_\_exp)
- long double **std::frexp** (long double \_\_x, int \* \_\_exp)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::hypot** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::hypot** (float \_\_x, float \_\_y)
- constexpr long double **std::hypot** (long double \_\_x, long double \_\_y)

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type std::ilogb (_Tp __x)`
- `constexpr int std::ilogb (float __x)`
- `constexpr int std::ilogb (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isfinite (_Tp)`
- `constexpr bool std::isfinite (double __x)`
- `constexpr bool std::isfinite (float __x)`
- `constexpr bool std::isfinite (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type std::isgreater (_Tp __x, _Up __y)`
- `constexpr bool std::isgreater (double __x, double __y)`
- `constexpr bool std::isgreater (float __x, float __y)`
- `constexpr bool std::isgreater (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type std::isgreaterequal (_Tp __x, _Up __y)`
- `constexpr bool std::isgreaterequal (double __x, double __y)`
- `constexpr bool std::isgreaterequal (float __x, float __y)`
- `constexpr bool std::isgreaterequal (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isinf (_Tp)`
- `constexpr bool std::isinf (double __x)`
- `constexpr bool std::isinf (float __x)`
- `constexpr bool std::isinf (long double __x)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type std::isless (_Tp __x, _Up __y)`
- `constexpr bool std::isless (double __x, double __y)`
- `constexpr bool std::isless (float __x, float __y)`
- `constexpr bool std::isless (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type std::islessequal (_Tp __x, _Up __y)`
- `constexpr bool std::islessequal (double __x, double __y)`
- `constexpr bool std::islessequal (float __x, float __y)`
- `constexpr bool std::islessequal (long double __x, long double __y)`
- `template<typename _Tp, typename _Up>`  
`constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type std::islessgreater (_Tp __x, _Up __y)`
- `constexpr bool std::islessgreater (double __x, double __y)`
- `constexpr bool std::islessgreater (float __x, float __y)`
- `constexpr bool std::islessgreater (long double __x, long double __y)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isnan (_Tp)`
- `constexpr bool std::isnan (double __x)`
- `constexpr bool std::isnan (float __x)`
- `constexpr bool std::isnan (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type std::isnormal (_Tp __x)`
- `constexpr bool std::isnormal (double __x)`

- constexpr bool **std::isnormal** (float \_\_x)
- constexpr bool **std::isnormal** (long double \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<(\_\_is\_arithmetic< \_Tp >::\_\_value && \_\_is\_arithmetic< \_Up >::\_\_value),  
bool >::\_\_type **std::isunordered** (\_Tp \_\_x, \_Up \_\_y)
- constexpr bool **std::isunordered** (double \_\_x, double \_\_y)
- constexpr bool **std::isunordered** (float \_\_x, float \_\_y)
- constexpr bool **std::isunordered** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::ldexp** (\_Tp \_\_x, int  
\_\_exp)
- constexpr float **std::ldexp** (float \_\_x, int \_\_exp)
- constexpr long double **std::ldexp** (long double \_\_x, int \_\_exp)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::lgamma** (\_Tp \_\_x)
- constexpr float **std::lgamma** (float \_\_x)
- constexpr long double **std::lgamma** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **std::llrint** (\_Tp \_\_x)
- constexpr long long **std::llrint** (float \_\_x)
- constexpr long long **std::llrint** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, longlong >::\_\_type **std::llround** (\_Tp \_\_x)
- constexpr long long **std::llround** (float \_\_x)
- constexpr long long **std::llround** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log** (\_Tp \_\_x)
- constexpr float **std::log** (float \_\_x)
- constexpr long double **std::log** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log10** (\_Tp \_\_x)
- constexpr float **std::log10** (float \_\_x)
- constexpr long double **std::log10** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log1p** (\_Tp \_\_x)
- constexpr float **std::log1p** (float \_\_x)
- constexpr long double **std::log1p** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log2** (\_Tp \_\_x)
- constexpr float **std::log2** (float \_\_x)
- constexpr long double **std::log2** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::logb** (\_Tp \_\_x)
- constexpr float **std::logb** (float \_\_x)
- constexpr long double **std::logb** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **std::lrint** (\_Tp \_\_x)
- constexpr long **std::lrint** (float \_\_x)
- constexpr long **std::lrint** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, long >::\_\_type **std::lround** (\_Tp \_\_x)
- constexpr long **std::lround** (float \_\_x)

- constexpr long **std::lround** (long double \_\_x)
- float **std::modf** (float \_\_x, float \*\_\_iptr)
- long double **std::modf** (long double \_\_x, long double \*\_\_iptr)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::nearbyint** (\_Tp \_\_x)
- constexpr float **std::nearbyint** (float \_\_x)
- constexpr long double **std::nearbyint** (long double \_\_x)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Up>::\_\_type **std::nextafter** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::nextafter** (float \_\_x, float \_\_y)
- constexpr long double **std::nextafter** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::nexttoward** (\_Tp \_\_x, long double \_\_y)
- constexpr float **std::nexttoward** (float \_\_x, long double \_\_y)
- constexpr long double **std::nexttoward** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Up>::\_\_type **std::pow** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::pow** (float \_\_x, float \_\_y)
- constexpr long double **std::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
constexpr \_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Up>::\_\_type **std::remainder** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::remainder** (float \_\_x, float \_\_y)
- constexpr long double **std::remainder** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up>  
\_\_gnu\_cxx::\_\_promote\_2<\_Tp, \_Up>::\_\_type **std::remquo** (\_Tp \_\_x, \_Up \_\_y, int \*\_\_pquo)
- float **std::remquo** (float \_\_x, float \_\_y, int \*\_\_pquo)
- long double **std::remquo** (long double \_\_x, long double \_\_y, int \*\_\_pquo)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::rint** (\_Tp \_\_x)
- constexpr float **std::rint** (float \_\_x)
- constexpr long double **std::rint** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::round** (\_Tp \_\_x)
- constexpr float **std::round** (float \_\_x)
- constexpr long double **std::round** (long double \_\_x)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::scalbn** (\_Tp \_\_x, long \_\_ex)
- constexpr float **std::scalbn** (float \_\_x, long \_\_ex)
- constexpr long double **std::scalbn** (long double \_\_x, long \_\_ex)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double >::\_\_type **std::scalbn** (\_Tp \_\_x, int \_\_ex)
- constexpr float **std::scalbn** (float \_\_x, int \_\_ex)
- constexpr long double **std::scalbn** (long double \_\_x, int \_\_ex)
- template<typename \_Tp>  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, bool >::\_\_type **std::signbit** (\_Tp \_\_x)
- constexpr bool **std::signbit** (double \_\_x)
- constexpr bool **std::signbit** (float \_\_x)
- constexpr bool **std::signbit** (long double \_\_x)

- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::sin (_Tp __x)`
- `constexpr float std::sin (float __x)`
- `constexpr long double std::sin (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::sinh (_Tp __x)`
- `constexpr float std::sinh (float __x)`
- `constexpr long double std::sinh (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::sqrt (_Tp __x)`
- `constexpr float std::sqrt (float __x)`
- `constexpr long double std::sqrt (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tan (_Tp __x)`
- `constexpr float std::tan (float __x)`
- `constexpr long double std::tan (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tanh (_Tp __x)`
- `constexpr float std::tanh (float __x)`
- `constexpr long double std::tanh (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::tgamma (_Tp __x)`
- `constexpr float std::tgamma (float __x)`
- `constexpr long double std::tgamma (long double __x)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::trunc (_Tp __x)`
- `constexpr float std::trunc (float __x)`
- `constexpr long double std::trunc (long double __x)`

### 7.232.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.233 ext/cmath File Reference

### Namespaces

- namespace [`\_\_gnu\_cxx`](#)

### Macros

- `#define _EXT_CMATH`

### Variables

- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants< _RealType >::__e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants< _RealType >::__gamma_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants< _RealType >::__ln_10`

- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__ln_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__ln_3`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__log10_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__log2_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_e`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__one_div_root_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__phi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_half`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_quarter`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__pi_third`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_3`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_5`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_7`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__root_pi_div_2`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__two_div_pi`
- `template<typename _RealType>`  
`constexpr _RealType __gnu_cxx::__math_constants<_RealType>::__two_div_root_pi`

### 7.233.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.234 tr1/cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CMATH`



## Functions

- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy>`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float std::tr1::betaf` (float \_\_x, float \_\_y)
- `long double std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_1f` (float \_\_k)
- `long double std::tr1::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (\_Tp \_\_k)
- `float std::tr1::comp_ellint_2f` (float \_\_k)
- `long double std::tr1::comp_ellint_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn>`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float std::tr1::comp_ellint_3f` (float \_\_k, float \_\_nu)
- `long double std::tr1::comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp>`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg` (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- `float std::tr1::conf_hypergf` (float \_\_a, float \_\_c, float \_\_x)
- `long double std::tr1::conf_hypergl` (long double \_\_a, long double \_\_c, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_if` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_bessel_kf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_bessel_kl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp>`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float std::tr1::cyl_neumannf` (float \_\_nu, float \_\_x)
- `long double std::tr1::cyl_neumannl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1` (\_Tp \_\_k, \_Tpp \_\_phi)
- `float std::tr1::ellint_1f` (float \_\_k, float \_\_phi)
- `long double std::tr1::ellint_1l` (long double \_\_k, long double \_\_phi)
- `template<typename _Tp, typename _Tpp>`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2` (\_Tp \_\_k, \_Tpp \_\_phi)

- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp>  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::fabs** (\_Tp \_\_x)
- float **std::tr1::fabs** (float \_\_x)
- long double **std::tr1::fabs** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp>  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **std::tr1::hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **std::tr1::hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp, typename \_Up>  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::tr1::pow** (\_Tp \_\_x, \_Up \_\_y)
- float **std::tr1::pow** (float \_\_x, float \_\_y)
- long double **std::tr1::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp>  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 7.234.1 Detailed Description

This is a TR1 C++ Library header.

## 7.235 codecvt File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CODECVT_SPECIALIZATION(_NAME, _ELEM)`
- `#define _GLIBCXX_CODECVT_SPECIALIZATION2(_NAME, _ELEM)`

### Enumerations

- enum `codecvt_mode` { `consume_header` , `generate_header` , `little_endian` }

### 7.235.1 Detailed Description

This is a Standard C++ Library header.

## 7.236 complex File Reference

### Classes

- class [std::complex< \\_Tp >](#)
- class [std::complex< double >](#)
- class [std::complex< float >](#)
- class [std::complex< long double >](#)

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp>`  
`_Tp std::__complex_abs (const complex< \_Tp > &__z)`
- `__complex__ double std::__complex_acos (__complex__ double __z)`
- `__complex__ float std::__complex_acos (__complex__ float __z)`
- `__complex__ long double std::__complex_acos (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< \_Tp > std::__complex_acos (const std::complex< \_Tp > &__z)`
- `__complex__ double std::__complex_acosh (__complex__ double __z)`
- `__complex__ float std::__complex_acosh (__complex__ float __z)`
- `__complex__ long double std::__complex_acosh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< \_Tp > std::__complex_acosh (const std::complex< \_Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::__complex_arg (const complex< \_Tp > &__z)`
- `__complex__ double std::__complex_asin (__complex__ double __z)`
- `__complex__ float std::__complex_asin (__complex__ float __z)`
- `__complex__ long double std::__complex_asin (const __complex__ long double &__z)`

- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_asinh (__complex__ double __z)`
- `__complex__ float std::__complex_asinh (__complex__ float __z)`
- `__complex__ long double std::__complex_asinh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_atan (__complex__ double __z)`
- `__complex__ float std::__complex_atan (__complex__ float __z)`
- `__complex__ long double std::__complex_atan (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__z)`
- `__complex__ double std::__complex_atanh (__complex__ double __z)`
- `__complex__ float std::__complex_atanh (__complex__ float __z)`
- `__complex__ long double std::__complex_atanh (const __complex__ long double &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `complex< double > std::__complex_proj (const complex< double > &__z)`
- `complex< float > std::__complex_proj (const complex< float > &__z)`
- `complex< long double > std::__complex_proj (const complex< long double > &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`

- `template<typename _Tp>`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::conj (_Tp __x)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp>`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp>`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `template<typename _Tp>`  
`_Tp constexpr std::norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`constexpr _Tp std::norm (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp>`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template ostream & std::operator<< (ostream &, const complex< double > &)`
- `template ostream & std::operator<< (ostream &, const complex< float > &)`
- `template ostream & std::operator<< (ostream &, const complex< long double > &)`
- `template wostream & std::operator<< (wostream &, const complex< double > &)`
- `template wostream & std::operator<< (wostream &, const complex< float > &)`
- `template wostream & std::operator<< (wostream &, const complex< long double > &)`
- `template<typename _Tp, typename _CharT, class _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`

- template `istream` & `std::operator>>` (`istream` &, `complex`< double > &)
- template `istream` & `std::operator>>` (`istream` &, `complex`< float > &)
- template `istream` & `std::operator>>` (`istream` &, `complex`< long double > &)
- template `wistream` & `std::operator>>` (`wistream` &, `complex`< double > &)
- template `wistream` & `std::operator>>` (`wistream` &, `complex`< float > &)
- template `wistream` & `std::operator>>` (`wistream` &, `complex`< long double > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::polar` (const `_Tp` &, const `_Tp` &=\_Tp(0))
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::pow` (const `_Tp` &, const `complex`< `_Tp` > &)
- template<typename `_Tp`, typename `_Up`>  
`std::complex`< typename `__gnu_cxx::__promote_2`< `_Tp`, `_Up` >::\_\_type > `std::pow` (const `_Tp` &\_\_x, const `std::complex`< `_Up` > &\_\_y)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::pow` (const `complex`< `_Tp` > &, const `_Tp` &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::pow` (const `complex`< `_Tp` > &, const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::pow` (const `complex`< `_Tp` > &, int)
- template<typename `_Tp`, typename `_Up`>  
`std::complex`< typename `__gnu_cxx::__promote_2`< `_Tp`, `_Up` >::\_\_type > `std::pow` (const `std::complex`< `_Tp` > &\_\_x, const `_Up` &\_\_y)
- template<typename `_Tp`, typename `_Up`>  
`std::complex`< typename `__gnu_cxx::__promote_2`< `_Tp`, `_Up` >::\_\_type > `std::pow` (const `std::complex`< `_Tp` > &\_\_x, const `std::complex`< `_Up` > &\_\_y)
- template<typename `_Tp`>  
`std::complex`< typename `__gnu_cxx::__promote`< `_Tp` >::\_\_type > `std::proj` (`_Tp` \_\_x)
- template<typename `_Tp`>  
`std::complex`< `_Tp` > `std::proj` (const `std::complex`< `_Tp` > &)
- template<typename `_Tp`>  
constexpr `__gnu_cxx::__promote`< `_Tp` >::\_\_type `std::real` (`_Tp` \_\_x)
- template<typename `_Tp`>  
constexpr `_Tp` `std::real` (const `complex`< `_Tp` > &\_\_z)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::sin` (const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::sinh` (const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::sqrt` (const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::tan` (const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
`complex`< `_Tp` > `std::tanh` (const `complex`< `_Tp` > &)
- template<typename `_Tp`>  
constexpr `complex`< `_Tp` > `std::operator+` (const `_Tp` &\_\_x, const `complex`< `_Tp` > &\_\_y)
- template<typename `_Tp`>  
constexpr `complex`< `_Tp` > `std::operator+` (const `complex`< `_Tp` > &\_\_x, const `_Tp` &\_\_y)
- template<typename `_Tp`>  
constexpr `complex`< `_Tp` > `std::operator+` (const `complex`< `_Tp` > &\_\_x, const `complex`< `_Tp` > &\_\_y)

- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator- (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator- (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator- (const complex<_Tp> &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator* (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator* (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator* (const complex<_Tp> &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator/ (const _Tp &__x, const complex<_Tp> &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator/ (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr complex<_Tp> std::operator/ (const complex<_Tp> &__x, const complex<_Tp> &__y)`
  
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex<_Tp> &__x, const _Tp &__y)`
- `template<typename _Tp>`  
`constexpr bool std::operator== (const complex<_Tp> &__x, const complex<_Tp> &__y)`

### 7.236.1 Detailed Description

This is a Standard C++ Library header.

## 7.237 tr1/complex File Reference

### Namespaces

- namespace `std`
- namespace `std::tr1`

### Macros

- `#define _GLIBCXX_TR1_COMPLEX`

### Functions

- `template<typename _Tp>`  
`std::complex<_Tp> std::tr1::acos (const std::complex<_Tp> &)`
- `template<typename _Tp>`  
`std::complex<_Tp> std::tr1::acosh (const std::complex<_Tp> &)`
- `template<typename _Tp>`  
`_Tp std::tr1::arg (const complex<_Tp> &)`

- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &)`
- `template<typename _Tp>`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp>`  
`constexpr _Tp std::tr1::imag (const complex< _Tp > &__z)`
- `template<typename _Tp>`  
`_Tp constexpr std::tr1::norm (const complex< _Tp > &)`
- `template<typename _Tp>`  
`complex< _Tp > std::tr1::polar (const _Tp &, const _Tp &=_Tp(0))`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp>`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp, typename _Up>`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp>`  
`constexpr _Tp std::tr1::real (const complex< _Tp > &__z)`

### 7.237.1 Detailed Description

This is a TR1 C++ Library header.

## 7.238 complex.h File Reference

### 7.238.1 Detailed Description

This is a Standard C++ Library header.

## 7.239 Concepts

Here is a list of all documented concepts with brief descriptions:



|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| <a href="#"><code>std::assignable_from</code></a>                     |     |
| [concept.assignable], concept assignable_from                         | 741 |
| <a href="#"><code>std::common_reference_with</code></a>               |     |
| [concept.commonref], concept common_reference_with                    | 741 |
| <a href="#"><code>std::common_with</code></a>                         |     |
| [concept.common], concept common_with                                 | 742 |
| <a href="#"><code>std::constructible_from</code></a>                  |     |
| [concept.constructible], concept constructible_from                   | 742 |
| <a href="#"><code>std::convertible_to</code></a>                      |     |
| [concept.convertible], concept convertible_to                         | 742 |
| <a href="#"><code>std::copy_constructible</code></a>                  |     |
| [concept.copyconstructible], concept copy_constructible               | 743 |
| <a href="#"><code>std::default_initializable</code></a>               |     |
| [concept.defaultinitializable], concept default_initializable         | 743 |
| <a href="#"><code>std::derived_from</code></a>                        |     |
| [concept.derived], concept derived_from                               | 743 |
| <a href="#"><code>std::destructible</code></a>                        |     |
| [concept.destructible], concept destructible                          | 743 |
| <a href="#"><code>std::equivalence_relation</code></a>                |     |
| [concept.equiv], concept equivalence_relation                         | 744 |
| <a href="#"><code>std::indirectly_comparable</code></a>               |     |
| [alg.req.ind.cmp], concept indirectly_comparable                      | 744 |
| <a href="#"><code>std::indirectly_copyable</code></a>                 |     |
| [alg.req.ind.copy], concept indirectly_copyable                       | 744 |
| <a href="#"><code>std::indirectly_movable</code></a>                  |     |
| [alg.req.ind.move], concept indirectly_movable                        | 744 |
| <a href="#"><code>std::indirectly_readable</code></a>                 |     |
| Requirements for types that are readable by applying operator*        | 744 |
| <a href="#"><code>std::indirectly_swappable</code></a>                |     |
| [alg.req.ind.swap], concept indirectly_swappable                      | 745 |
| <a href="#"><code>std::indirectly_writable</code></a>                 |     |
| Requirements for writing a value into an iterator's referenced object | 745 |
| <a href="#"><code>std::invocable</code></a>                           |     |
| [concept.invocable], concept invocable                                | 745 |
| <a href="#"><code>std::mergeable</code></a>                           |     |
| [alg.req.mergeable], concept mergeable                                | 746 |
| <a href="#"><code>std::move_constructible</code></a>                  |     |
| [concept.moveconstructible], concept move_constructible               | 746 |

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| <a href="#">std::permutable</a>                                      |     |
| [alg.req.permutable], concept permutable                             | 746 |
| <a href="#">std::predicate</a>                                       |     |
| [concept.predicate], concept predicate                               | 746 |
| <a href="#">std::ranges::bidirectional_range</a>                     |     |
| A range for which ranges::begin returns a bidirectional iterator     | 747 |
| <a href="#">std::ranges::borrowed_range</a>                          |     |
| [range.range] The borrowed_range concept                             | 747 |
| <a href="#">std::ranges::common_range</a>                            |     |
| A range for which ranges::begin and ranges::end return the same type | 747 |
| <a href="#">std::ranges::contiguous_range</a>                        |     |
| A range for which ranges::begin returns a contiguous iterator        | 747 |
| <a href="#">std::ranges::forward_range</a>                           |     |
| A range for which ranges::begin returns a forward iterator           | 747 |
| <a href="#">std::ranges::input_range</a>                             |     |
| A range for which ranges::begin returns an input iterator            | 748 |
| <a href="#">std::ranges::output_range</a>                            |     |
| A range for which ranges::begin returns an output iterator           | 748 |
| <a href="#">std::ranges::random_access_range</a>                     |     |
| A range for which ranges::begin returns a random access iterator     | 748 |
| <a href="#">std::ranges::range</a>                                   |     |
| [range.range] The range concept                                      | 748 |
| <a href="#">std::ranges::sized_range</a>                             |     |
| [range.sized] The sized_range concept                                | 749 |
| <a href="#">std::ranges::view</a>                                    |     |
| [range.view] The ranges::view concept                                | 749 |
| <a href="#">std::ranges::viewable_range</a>                          |     |
| A range which can be safely converted to a view                      | 749 |
| <a href="#">std::regular_invocable</a>                               |     |
| [concept.regularinvocable], concept regular_invocable                | 749 |
| <a href="#">std::relation</a>                                        |     |
| [concept.relation], concept relation                                 | 750 |
| <a href="#">std::same_as</a>                                         |     |
| [concept.same], concept same_as                                      | 750 |
| <a href="#">std::simd::__vec_builtin</a>                             | 750 |
| <a href="#">std::simd::__vec_builtin_of</a>                          | 750 |
| <a href="#">std::sortable</a>                                        |     |
| [alg.req.sortable], concept sortable                                 | 751 |

|                                                                      |                     |
|----------------------------------------------------------------------|---------------------|
| <a href="#">std::strict_weak_order</a>                               |                     |
| [concept.strictweakorder], concept <a href="#">strict_weak_order</a> | <a href="#">751</a> |
| <a href="#">std::uniform_random_bit_generator</a>                    |                     |
| Requirements for a uniform random bit generator                      | <a href="#">751</a> |
| <a href="#">std::weakly_incrementable</a>                            |                     |
| Requirements on types that can be incremented with ++                | <a href="#">752</a> |

## 7.240 condition\_variable File Reference

### Classes

- class [std::condition\\_variable](#)
- class [std::condition\\_variable\\_any](#)

### Namespaces

- namespace [std](#)

### Enumerations

- enum class [std::cv\\_status](#) { [no\\_timeout](#) , [timeout](#) }

### Functions

- void [std::notify\\_all\\_at\\_thread\\_exit](#) ([condition\\_variable](#) &, [unique\\_lock](#)< [mutex](#) >)

#### 7.240.1 Detailed Description

This is a Standard C++ Library header.

## 7.241 csetjmp File Reference

### Namespaces

- namespace [std](#)

### Macros

- #define [\\_GLIBCXX\\_CSETJMP](#)
- #define [setjmp](#)(env)

#### 7.241.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.242 csignal File Reference

### Namespaces

- namespace [std](#)

**Macros**

- `#define _GLIBCXX_CSIGNAL`

**7.242.1 Detailed Description**

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

**7.243 cstdalign File Reference****Macros**

- `#define _GLIBCXX_CSTDALIGN`

**7.243.1 Detailed Description**

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

**7.244 cstdarg File Reference****Namespaces**

- namespace [std](#)

**Macros**

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

**7.244.1 Detailed Description**

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

**7.245 tr1/cstdarg File Reference****Macros**

- `#define _GLIBCXX_TR1_CSTDARG`

**7.245.1 Detailed Description**

This is a TR1 C++ Library header.

**7.246 cstdbool File Reference****Macros**

- `#define _GLIBCXX_CSTDBOOL`

### 7.246.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 7.247 tr1/cstdbool File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDBOOL`

### 7.247.1 Detailed Description

This is a TR1 C++ Library header.

## 7.248 cstdint File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDDEF`

### 7.248.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdint.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.249 cstdint File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTDINT`

### 7.249.1 Detailed Description

This is a Standard C++ Library header.

## 7.250 tr1/cstdint File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

## Macros

- `#define _GLIBCXX_TR1_CSTDINT`

### 7.250.1 Detailed Description

This is a TR1 C++ Library header.

## 7.251 cstdio File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CSTDIO`

### 7.251.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.252 tr1/cstdio File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDIO`

### 7.252.1 Detailed Description

This is a TR1 C++ Library header.

## 7.253 cstdlib File Reference

### Namespaces

- namespace [std](#)

## Macros

- `#define _GLIBCXX_CSTDLIB`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

## Functions

- `ldiv_t std::div (long __i, long __j) noexcept`

### 7.253.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.254 tr1/cstdlib File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDLIB`

#### 7.254.1 Detailed Description

This is a TR1 C++ Library header.

## 7.255 cstring File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CSTRING`

### Functions

- `void * std::memchr (void *__s, int __c, size_t __n)`
- `char * std::strchr (char *__s, int __n)`
- `char * std::strpbrk (char *__s1, const char *__s2)`
- `char * std::strrchr (char *__s, int __n)`
- `char * std::strstr (char *__s1, const char *__s2)`

#### 7.255.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.256 ctgmath File Reference

### Macros

- `#define _GLIBCXX_CTGMATH`

#### 7.256.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

## 7.257 tr1/ctgmath File Reference

### Macros

- `#define _GLIBCXX_TR1_CTGMATH`

#### 7.257.1 Detailed Description

This is a TR1 C++ Library header.

## 7.258 ctime File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CTIME`

#### 7.258.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.259 tr1/ctime File Reference

### Macros

- `#define _GLIBCXX_TR1_CTIME`

#### 7.259.1 Detailed Description

This is a TR1 C++ Library header.

## 7.260 cuchar File Reference

### Macros

- `#define _GLIBCXX_CUCHAR`

#### 7.260.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.261 wchar File Reference

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_CWCHAR`

### Functions

- `wchar_t * std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`



### 7.261.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.262 tr1/cwchar File Reference

### Namespaces

- namespace `std`
- namespace `std::tr1`

### Macros

- `#define _GLIBCXX_TR1_CWCHAR`

### 7.262.1 Detailed Description

This is a TR1 C++ Library header.

## 7.263 cwctype File Reference

### Namespaces

- namespace `std`

### Macros

- `#define _GLIBCXX_CWCTYPE`

### 7.263.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 7.264 tr1/cwctype File Reference

### Namespaces

- namespace `std`
- namespace `std::tr1`

### Macros

- `#define _GLIBCXX_TR1_CWCTYPE`

### 7.264.1 Detailed Description

This is a TR1 C++ Library header.

## 7.265 assertions.h File Reference

### Macros

- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_subscript(_N)`
- `#define __GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define __GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define __GLIBCXX_DEBUG_PEDASSERT(_Condition)`

### 7.265.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.266 debug.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define __glibcxx_requires_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_can_increment(_First, _Size)`
- `#define __glibcxx_requires_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive(_First, _Last)`
- `#define __glibcxx_requires_irreflexive2(_First, _Last)`
- `#define __glibcxx_requires_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`

### 7.266.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.267 formatter.h File Reference

### Classes

- class [\\_\\_gnu\\_debug::type\\_info](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Macros

- `#define _GLIBCXX_TYPEID(_Type)`

### Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move`↵  
`__assign`,  
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range`↵  
`from_self`,  
`__msg_irreflexive_ordering` }

### Functions

- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__check_singular (_Iterator const &)`

#### 7.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.268 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`

- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred>`  
`constexpr bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value, _Pred __pred)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last, ↵`  
`Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, ↵`  
`Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _ForwardIterator>`  
`constexpr bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last,`  
`std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate,`  
`std::input\_iterator\_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &,`  
`std::input\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`constexpr bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__↵`  
`last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__↵`  
`last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, ↵`  
`Predicate, std::\_\_false\_type)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::↵`  
`\_\_false\_type)`
- `template<typename _InputIterator, typename _Predicate>`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__↵`  
`last, _Predicate __pred, std::\_\_true\_type)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__↵`  
`last, std::\_\_true\_type)`
- `template<typename _InputIterator>`  
`constexpr _InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__first, const _InputIterator`  
`&__last, const char *__file, unsigned int __line, const char *__function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>`  
`bool __gnu_debug::__foreign_iterator (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &__it, ↵`  
`_InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename`  
`_OtherCategory>`  
`bool __gnu_debug::__foreign_iterator_aux2 (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const`

- [\\_Safe\\_iterator](#)< \_OtherIterator, \_OtherSequence, \_OtherCategory > &, const [\\_Safe\\_iterator](#)< \_OtherIterator, \_OtherSequence, \_OtherCategory > &)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_InputIterator>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux2](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &\_\_it, const \_InputIterator &\_\_other, const \_InputIterator &\_\_other\_end)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_OtherIterator>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux2](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &\_\_it, const [\\_Safe\\_iterator](#)< \_OtherIterator, \_Sequence, \_Category > &\_\_other, const [\\_Safe\\_iterator](#)< \_OtherIterator, \_Sequence, \_Category > &)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_InputIterator>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux3](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &, const \_InputIterator &, const \_InputIterator &, std::\_\_false\_type)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_InputIterator>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux3](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &\_\_it, const \_InputIterator &\_\_other, const \_InputIterator &\_\_other\_end, std::\_\_true\_type)
- template<typename \_Iterator, typename \_Sequence, typename \_Category>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux4](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &,...)
- template<typename \_Iterator, typename \_Sequence, typename \_Category>  
bool [\\_\\_gnu\\_debug::\\_\\_foreign\\_iterator\\_aux4](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &\_\_it, const typename \_Sequence::value\_type \*\_\_other)
- template<typename \_Iterator>  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_is\\_irreflexive](#) (\_Iterator \_\_it)
- template<typename \_Iterator, typename \_Pred>  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_is\\_irreflexive\\_pred](#) (\_Iterator \_\_it, \_Pred \_\_pred)

### 7.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.269 helper\_functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Enumerations

- enum [\\_\\_gnu\\_debug::\\_\\_Distance\\_precision](#) {  
    [\\_\\_dp\\_none](#) , [\\_\\_dp\\_equality](#) , [\\_\\_dp\\_sign](#) , [\\_\\_dp\\_sign\\_max\\_size](#) ,  
    [\\_\\_dp\\_exact](#) }

### Functions

- template<typename \_Iterator>  
constexpr \_Iterator [\\_\\_gnu\\_debug::\\_\\_base](#) (\_Iterator \_\_it)
- template<typename \_InputIterator, typename \_Size>  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_can\\_advance](#) (\_InputIterator, \_Size)
- template<typename \_InputIterator, typename \_Diff>  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_can\\_advance](#) (\_InputIterator, const std::pair< \_Diff, [\\_Distance\\_precision](#) > &, int)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_Size>  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_can\\_advance](#) (const [\\_Safe\\_iterator](#)< \_Iterator, \_Sequence, \_Category > &, \_Size)

- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const __Safe_iterator< _Iterator, _Sequence, _Category > &,`  
`const std::pair< _Diff, _Distance_precision > &, int)`
- `template<typename _Iterator>`  
`constexpr bool __gnu_debug::__check_singular (_Iterator const &)`
- `template<typename _Tp>`  
`constexpr bool __gnu_debug::__check_singular (_Tp *const &__ptr)`
- `bool __gnu_debug::__check_singular_aux (const class __Safe_iterator_base *)`
- `bool __gnu_debug::__check_singular_aux (const void *)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (_Iterator __lhs, _Iterator __↵`  
`rhs)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (_Iterator __lhs, _Iterator __↵`  
`rhs, std::input_iterator_tag)`
- `template<typename _Iterator>`  
`constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (_Iterator __lhs, _Iterator __rhs,`  
`std::random_access_iterator_tag)`
- `template<typename _Iterator>`  
`constexpr _Iterator __gnu_debug::__unsafe (_Iterator __it)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__valid_range (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__valid_range (_InputIterator __first, _InputIterator __last, typename __↵`  
`Distance_traits< _InputIterator >::__type &__dist)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __gnu_debug::__valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &,`  
`const __Safe_iterator< _Iterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __gnu_debug::__valid_range (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const`  
`__Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __gnu_debug::__valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator<`  
`__Iterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence>`  
`bool __gnu_debug::__valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &, const __Safe_local_iterator<`  
`__Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__valid_range_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _InputIterator>`  
`constexpr bool __gnu_debug::__valid_range_aux (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)`

### 7.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.270 macros.h File Reference

### Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_can_increment(_First, _Size)`

- `#define __glibcxx_check_can_increment_dist(_First, _Dist, _Way)`
- `#define __glibcxx_check_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_equal_allocs(_This, _Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase2(_CPosition)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_irreflexive(_First, _Last)`
- `#define __glibcxx_check_irreflexive2(_First, _Last)`
- `#define __glibcxx_check_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_constructor_range(_First, _Last)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __glibcxx_check_valid_range2(_First, _Last, _Dist)`
- `#define __glibcxx_check_valid_range_at(_First, _Last, _File, _Line, _Func)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Cond, _ErrMsg)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT(_Cond, _ErrMsg, _File, _Line)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT_F(_Cond, _ErrMsg, _File, _Line, _Func)`

### 7.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 7.270.2 Macro Definition Documentation

#### `__glibcxx_check_erase`

```
#define __glibcxx_check_erase(
 _Position)
```

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

**\_\_glibcxx\_check\_erase\_after**

```
#define __glibcxx_check_erase_after(
 _Position)
```

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

**\_\_glibcxx\_check\_erase\_range**

```
#define __glibcxx_check_erase_range(
 _First,
 _Last)
```

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

**\_\_glibcxx\_check\_erase\_range\_after**

```
#define __glibcxx_check_erase_range_after(
 _First,
 _Last)
```

Verify that we can erase the elements in the iterator range `(_First, _Last)`. We can erase the elements if `(_First, _Last)` is a valid iterator range within this sequence.

**\_\_glibcxx\_check\_heap\_pred**

```
#define __glibcxx_check_heap_pred(
 _First,
 _Last,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

**\_\_glibcxx\_check\_insert**

```
#define __glibcxx_check_insert(
 _Position)
```

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

**\_\_glibcxx\_check\_insert\_after**

```
#define __glibcxx_check_insert_after(
 _Position)
```

Verify that we can insert into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_↵` iterator.

**\_\_glibcxx\_check\_insert\_range**

```
#define __glibcxx_check_insert_range(
 _Position,
 _First,
 _Last,
 _Dist)
```



Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_insert_range_after`**

```
#define __glibcxx_check_insert_range_after(
 _Position,
 _First,
 _Last,
 _Dist)
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

#### **`__glibcxx_check_partitioned_lower`**

```
#define __glibcxx_check_partitioned_lower(
 _First,
 _Last,
 _Value)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

#### **`__glibcxx_check_partitioned_lower_pred`**

```
#define __glibcxx_check_partitioned_lower_pred(
 _First,
 _Last,
 _Value,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

#### **`__glibcxx_check_partitioned_upper_pred`**

```
#define __glibcxx_check_partitioned_upper_pred(
 _First,
 _Last,
 _Value,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

#### **`__glibcxx_check_sorted_pred`**

```
#define __glibcxx_check_sorted_pred(
 _First,
 _Last,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

## **`_GLIBCXX_DEBUG_VERIFY_AT_F`**

```
#define _GLIBCXX_DEBUG_VERIFY_AT_F(
 _Cond,
 _ErrMsg,
 _File,
 _Line,
 _Func)
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

## **7.271 map.h File Reference**

### **Classes**

- class `std::__debug::map<_Key, _Tp, _Compare, _Allocator>`

### **Namespaces**

- namespace `std`
- namespace `std::__debug`

### **Functions**

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::map** (`_InputIterator, _InputIterator, _Allocator`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, less<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_val_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::map** (`_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()`) -> `map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::map** (`initializer_list<pair<_Key, _Tp>>, _Allocator`) -> `map<_Key, _Tp, less<_Key>, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::map** (`initializer_list<pair<_Key, _Tp>>, _Compare=_Compare(), _Allocator=_Allocator()`) -> `map<_Key, _Tp, _Compare, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t<pair<const _Key, _Tp>> std::__debug::operator<=>` (`const map<_Key, _Tp, _Compare, _Alloc> &__lhs, const map<_Key, _Tp, _Compare, _Alloc> &__rhs`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator==` (`const map<_Key, _Tp, _Compare, _Allocator> &__lhs, const map<_Key, _Tp, _Compare, _Allocator> &__rhs`)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>`  
`void std::__debug::swap` (`map<_Key, _Tp, _Compare, _Allocator> &__lhs, map<_Key, _Tp, _Compare, _Allocator> &__rhs`) `noexcept(/*conditional*/)`

### **7.271.1 Detailed Description**

This file is a GNU debug extension to the Standard C++ Library.

## 7.272 multimap.h File Reference

### Classes

- class `std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- template<typename \_InputIterator, typename \_Allocator, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireAllocator<\_Allocator>>  
**std::\_\_debug::multimap** (\_InputIterator, \_InputIterator, \_Allocator) -> multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, less< \_\_iter\_key\_t< \_InputIterator > >, \_Allocator >
- template<typename \_InputIterator, typename \_Compare = less< \_\_iter\_key\_t< \_InputIterator > >, typename \_Allocator = allocator< \_\_iter\_key\_t< \_InputIterator > >, typename = \_RequireInputIter<\_InputIterator>, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>  
**std::\_\_debug::multimap** (\_InputIterator, \_InputIterator, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multimap< \_\_iter\_key\_t< \_InputIterator >, \_\_iter\_val\_t< \_InputIterator >, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Allocator, typename = \_RequireAllocator<\_Allocator>>  
**std::\_\_debug::multimap** (initializer\_list< pair< \_Key, \_Tp > >, \_Allocator) -> multimap< \_Key, \_Tp, less< \_Key >, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Compare = less< \_Key >, typename \_Allocator = allocator< pair< const \_Key, \_Tp > >, typename = \_RequireNotAllocator<\_Compare>, typename = \_RequireAllocator<\_Allocator>>  
**std::\_\_debug::multimap** (initializer\_list< pair< \_Key, \_Tp > >, \_Compare=\_Compare(), \_Allocator=\_Allocator()) -> multimap< \_Key, \_Tp, \_Compare, \_Allocator >
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Alloc>  
\_\_detail::\_\_synth3way\_t< pair< const \_Key, \_Tp > > **std::\_\_debug::operator<=>** (const multimap< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_lhs, const multimap< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_rhs)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Allocator>  
bool **std::\_\_debug::operator==** (const multimap< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, const multimap< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs)
- template<typename \_Key, typename \_Tp, typename \_Compare, typename \_Allocator>  
void **std::\_\_debug::swap** (multimap< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_lhs, multimap< \_Key, \_Tp, \_Compare, \_Allocator > &\_\_rhs) noexcept(*/\*conditional\*/*)

#### 7.272.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.273 multiset.h File Reference

### Classes

- class `std::__debug::multiset<_Key, _Compare, _Allocator>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

## Functions

- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator\_traits< _InputIterator >::value_type, less< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator\_traits< _InputIterator >::value_type, _Compare, _Allocator >`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>`  
`std::__debug::multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >`
- `template<typename _Key, typename _Compare, typename _Alloc>`  
`__detail::__synth3way_t< _Key > std::__debug::operator<=> (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>`  
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/\*conditional \*/)`

### 7.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.274 [safe\\_base.h](#) File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence\\_base](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const [\\_Safe\\_iterator\\_base](#) \* \_\_x)

### 7.274.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.275 [safe\\_container.h](#) File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_container<\\_SafeContainer, \\_Alloc, \\_SafeBase, \\_IsCxx11AllocatorAware >](#)

## Namespaces

- namespace `__gnu_debug`

### 7.275.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.276 `safe_iterator.h` File Reference

### Classes

- struct `__gnu_debug::BeforeBeginHelper<_Sequence>`
- class `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>`
- struct `__gnu_debug::Sequence_traits<_Sequence>`

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_BEGIN`
- `#define _GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_END`
- `#define _GLIBCXX_DEBUG_VERIFY_DIST_OPERANDS(_Lhs, _Rhs)`
- `#define _GLIBCXX_DEBUG_VERIFY_EQ_OPERANDS(_Lhs, _Rhs)`
- `#define _GLIBCXX_DEBUG_VERIFY_OPERANDS(_Lhs, _Rhs, _BadMsgId, _DiffMsgId)`
- `#define _GLIBCXX_DEBUG_VERIFY_REL_OPERANDS(_Lhs, _Rhs)`

### Functions

- `template<typename _Iterator, typename _Sequence>`  
`constexpr _Iterator __gnu_debug::__base (const _Safe_iterator<_Iterator, _Sequence, std::random_access_iterator_tag> & __it)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>`  
`constexpr bool __gnu_debug::__can_advance (const _Safe_iterator<_Iterator, _Sequence, _Category> &, _Size)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>`  
`constexpr bool __gnu_debug::__can_advance (const _Safe_iterator<_Iterator, _Sequence, _Category> &, const std::pair<_Diff, _Distance_precision> &, int)`
- `template<typename _Iterator, typename _Sequence>`  
`_Iterator __gnu_debug::__unsafe (_Safe_iterator<_Iterator, _Sequence> & __it)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __gnu_debug::__valid_range (const _Safe_iterator<_Iterator, _Sequence, _Category> &, const _Safe_iterator<_Iterator, _Sequence, _Category> &)`
- `template<typename _Iterator, typename _Sequence, typename _Category>`  
`constexpr bool __gnu_debug::__valid_range (const _Safe_iterator<_Iterator, _Sequence, _Category> &, const _Safe_iterator<_Iterator, _Sequence, _Category> &, typename _Distance_traits<_Iterator>::__type &)`

### 7.276.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.277 safe\_iterator.tcc File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)

### Functions

- `template<bool _IsMove, typename _I1, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::copy_move_a (_I1, _I1, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _Olte, typename _OSeq, typename _OCat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > std::copy_move_a (const \_\_gnu\_debug::Safe\_iterator<`  
`_Ite, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Ite, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator<`  
`_Olte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::copy_move_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _I1, typename _Ite, typename _Seq, typename _Cat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::copy_move_backward_a (_I1, _I1, const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _Olte, typename _OSeq, typename _OCat>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > std::copy_move_backward_a (const`  
`\_\_gnu\_debug::Safe\_iterator< _Ite, _ISeq, _ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Ite, _ISeq, ↵`  
`_ICat > &, const \_\_gnu\_debug::Safe\_iterator< _Olte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>`  
`constexpr _OI std::copy_move_backward_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &,`  
`const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<typename _I1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::equal_aux (_I1, _I1, const \_\_gnu\_debug::Safe\_iterator< _I2, _Seq2, _Cat2 > &)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::equal_aux (const \_\_gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, _I2)`
- `template<typename _I1, typename _Seq1, typename _Cat1, typename _I2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::equal_aux (const \_\_gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const`  
`\_\_gnu\_debug::Safe\_iterator< _I1, _Seq1, _Cat1 > &, const \_\_gnu\_debug::Safe\_iterator< _I2, _Seq2,`  
`_Cat2 > &)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>`  
`constexpr void std::fill_a (const \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > &, const \_\_gnu\_debug::Safe\_iterator<`  
`_Ite, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>`  
`constexpr \_\_gnu\_debug::Safe\_iterator< _Ite, _Seq, _Cat > std::fill_n_a (const \_\_gnu\_debug::Safe\_iterator<`  
`_Ite, _Seq, _Cat > & __first, _Size __n, const _Tp & __value, std::input\_iterator\_tag)`
- `template<typename _I1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::lexicographical_compare_aux (_I1 __first1, _I1 __last1, const \_\_gnu\_debug::Safe\_iterator<`  
`_Ite2, _Seq2, _Cat2 > & __first2, const \_\_gnu\_debug::Safe\_iterator< _Ite2, _Seq2, _Cat2 > & __last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _I2>`  
`constexpr bool std::lexicographical_compare_aux (const \_\_gnu\_debug::Safe\_iterator< _Ite1, _Seq1, ↵`  
`_Cat1 > & __first1, const \_\_gnu\_debug::Safe\_iterator< _Ite1, _Seq1, _Cat1 > & __last1, _I2 __first2, _I2`  
`__last2)`
- `template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>`  
`constexpr bool std::lexicographical_compare_aux (const \_\_gnu\_debug::Safe\_iterator< _Ite1, ↵`  
`_Seq1, _Cat1 > & __first1, const \_\_gnu\_debug::Safe\_iterator< _Ite1, _Seq1, _Cat1 > & __last1, const`

```

::__gnu_debug::__Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::__Safe_iterator< ↵
_Ite2, _Seq2, _Cat2 > &__last2)

```

- `template<typename _Ite, typename _Seq>`  
`constexpr decltype(std::__niter_base(std::declval<_Ite>())) std::__niter_base (const ::__gnu_debug::__Safe_iterator<`  
`_Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible<_Ite>::↵`  
`value)`

### 7.277.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.278 `safe_local_iterator.h` File Reference

### Classes

- class `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`

### Namespaces

- namespace `__gnu_debug`

### Macros

- `#define _GLIBCXX_DEBUG_VERIFY_OPERANDS(_Lhs, _Rhs)`

### Functions

- `template<typename _Iterator, typename _UContainer>`  
`_Iterator __gnu_debug::__unsafe (const __Safe_local_iterator< _Iterator, _UContainer > &__it)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __gnu_debug::__valid_range (const __Safe_local_iterator< _Iterator, _UContainer > &__first, const`  
`__Safe_local_iterator< _Iterator, _UContainer > &__last)`
- `template<typename _Iterator, typename _UContainer>`  
`bool __gnu_debug::__valid_range (const __Safe_local_iterator< _Iterator, _UContainer > &__first, const`  
`__Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__↵`  
`dist_info)`

### 7.278.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.279 `safe_local_iterator.tcc` File Reference

### Namespaces

- namespace `__gnu_debug`

### 7.279.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.280 `safe_sequence.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::After\\_nth\\_from<\\_Iterator>](#)
- class [\\_\\_gnu\\_debug::Equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::Not\\_equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::Safe\\_node\\_sequence<\\_Sequence>](#)
- class [\\_\\_gnu\\_debug::Safe\\_sequence<\\_Sequence>](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

#### 7.280.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.281 `safe_sequence.tcc` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

#### 7.281.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.282 `safe_unordered_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::Safe\\_local\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::Safe\\_unordered\\_container\\_base](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

#### 7.282.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.283 `safe_unordered_container.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::Safe\\_unordered\\_container<\\_Container>](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

#### 7.283.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.



## 7.284 `safe_unordered_container.tcc` File Reference

### Namespaces

- namespace `__gnu_debug`

#### 7.284.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.285 `set.h` File Reference

### Classes

- class `std::__debug::set<_Key, _Compare, _Allocator>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Functions

- `template<typename _Key, typename _Compare, typename _Alloc>  
__detail::__synth3way_t<_Key> std::__debug::operator<=> (const set<_Key, _Compare, _Alloc> &__lhs,  
const set<_Key, _Compare, _Alloc> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>  
bool std::__debug::operator== (const set<_Key, _Compare, _Allocator> &__lhs, const set<_Key, _Compare,  
_Allocator> &__rhs)`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllo-  
cator<_Allocator>>  
std::__debug::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits<_InputIterator>::value_type, less< typename iterator_traits<_InputIterator>::value_type>, _Allocator>`
- `template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allo-  
cator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename =  
_RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>  
std::__debug::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< type-  
name iterator_traits<_InputIterator>::value_type, _Compare, _Allocator>`
- `template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>  
std::__debug::set (initializer_list<_Key>, _Allocator) -> set<_Key, less<_Key>, _Allocator>`
- `template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllo-  
cator<_Compare>, typename = _RequireAllocator<_Allocator>>  
std::__debug::set (initializer_list<_Key>, _Compare=_Compare(), _Allocator=_Allocator()) -> set<_Key, _Compare, _Allocator>`
- `template<typename _Key, typename _Compare, typename _Allocator>  
void std::__debug::swap (set<_Key, _Compare, _Allocator> &__x, set<_Key, _Compare, _Allocator> &__y)  
noexcept(/*conditional */)`

#### 7.285.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.286 `debugging` File Reference

#### 7.286.1 Detailed Description

This is a Standard C++ Library header.

## 7.287 decimal File Reference

### Classes

- class [std::decimal::decimal128](#)
- class [std::decimal::decimal32](#)
- class [std::decimal::decimal64](#)

### Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

### Macros

- `#define _DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_DEC(_Op, _T1, _T2, _T3)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_INT(_Op, _Tp)`
- `#define _DECLARE_DECIMAL_COMPARISON(_Op, _Tp)`
- `#define _GLIBCXX_DECIMAL`
- `#define _GLIBCXX_USE_DECIMAL_`

### Functions

- double [std::decimal::decimal128\\_to\\_double](#) ([decimal128](#) \_\_d)
- float [std::decimal::decimal128\\_to\\_float](#) ([decimal128](#) \_\_d)
- long double [std::decimal::decimal128\\_to\\_long\\_double](#) ([decimal128](#) \_\_d)
- long long [std::decimal::decimal128\\_to\\_long\\_long](#) ([decimal128](#) \_\_d)
- double [std::decimal::decimal32\\_to\\_double](#) ([decimal32](#) \_\_d)
- float [std::decimal::decimal32\\_to\\_float](#) ([decimal32](#) \_\_d)
- long double [std::decimal::decimal32\\_to\\_long\\_double](#) ([decimal32](#) \_\_d)
- long long [std::decimal::decimal32\\_to\\_long\\_long](#) ([decimal32](#) \_\_d)
- double [std::decimal::decimal64\\_to\\_double](#) ([decimal64](#) \_\_d)
- float [std::decimal::decimal64\\_to\\_float](#) ([decimal64](#) \_\_d)
- long double [std::decimal::decimal64\\_to\\_long\\_double](#) ([decimal64](#) \_\_d)
- long long [std::decimal::decimal64\\_to\\_long\\_long](#) ([decimal64](#) \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) ([decimal128](#) \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) ([decimal32](#) \_\_d)
- double [std::decimal::decimal\\_to\\_double](#) ([decimal64](#) \_\_d)
- float [std::decimal::decimal\\_to\\_float](#) ([decimal128](#) \_\_d)
- float [std::decimal::decimal\\_to\\_float](#) ([decimal32](#) \_\_d)
- float [std::decimal::decimal\\_to\\_float](#) ([decimal64](#) \_\_d)
- long double [std::decimal::decimal\\_to\\_long\\_double](#) ([decimal128](#) \_\_d)
- long double [std::decimal::decimal\\_to\\_long\\_double](#) ([decimal32](#) \_\_d)
- long double [std::decimal::decimal\\_to\\_long\\_double](#) ([decimal64](#) \_\_d)
- long long [std::decimal::decimal\\_to\\_long\\_long](#) ([decimal128](#) \_\_d)
- long long [std::decimal::decimal\\_to\\_long\\_long](#) ([decimal32](#) \_\_d)
- long long [std::decimal::decimal\\_to\\_long\\_long](#) ([decimal64](#) \_\_d)
- static [decimal128](#) [std::decimal::make\\_decimal128](#) (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) [std::decimal::make\\_decimal128](#) (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) [std::decimal::make\\_decimal32](#) (long long \_\_coeff, int \_\_exp)

- static [decimal32](#) `std::decimal::make_decimal32` (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) `std::decimal::make_decimal64` (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) `std::decimal::make_decimal64` (unsigned long long \_\_coeff, int \_\_exp)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, long long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, long long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- bool `std::decimal::operator!=` ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- bool `std::decimal::operator!=` (int \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (int \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (int \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` (long \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (long \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (long \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` (long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (long long \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool `std::decimal::operator!=` (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128](#) `std::decimal::operator*` ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128](#) `std::decimal::operator*` ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128](#) `std::decimal::operator*` ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)

- Generated by Doxygen

- Generated by Doxygen

- Generated by Doxygen



- `decimal32 std::decimal::operator/ (decimal32 __lhs, unsigned int __rhs)`
- `decimal32 std::decimal::operator/ (decimal32 __lhs, unsigned long __rhs)`
- `decimal32 std::decimal::operator/ (decimal32 __lhs, unsigned long long __rhs)`
- `decimal128 std::decimal::operator/ (decimal64 __lhs, decimal128 __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, int __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, long __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, long long __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, unsigned int __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, unsigned long __rhs)`
- `decimal64 std::decimal::operator/ (decimal64 __lhs, unsigned long long __rhs)`
- `decimal128 std::decimal::operator/ (int __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (int __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (int __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator/ (long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (long __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (long __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator/ (long long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (long long __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (long long __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator/ (unsigned int __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (unsigned int __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (unsigned int __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator/ (unsigned long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (unsigned long __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (unsigned long __lhs, decimal64 __rhs)`
- `decimal128 std::decimal::operator/ (unsigned long long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator/ (unsigned long long __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator/ (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal64 __rhs)`

- Generated by Doxygen



- bool **std::decimal::operator==** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator>** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>** (long \_\_lhs, decimal64 \_\_rhs)

- Generated by Doxygen

- bool **std::decimal::operator>=** (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)

### 7.287.1 Detailed Description

This is a Standard C++ Library header.

## 7.288 debug/deque File Reference

### Classes

- class [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_DEQUE`

### Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::deque** (\_InputIterator, \_InputIterator, \_Allocator=\_Allocator()) -> deque< \_ValT, \_Allocator >
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>`  
**std::\_\_debug::deque** (size\_t, \_Tp, \_Allocator=\_Allocator()) -> deque< \_Tp, \_Allocator >
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const deque< _Tp, _Alloc > &__x,`  
`const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional`  
`*/)`

### 7.288.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.289 deque File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Tp>`  
`using std::pmr::deque`

### 7.289.1 Detailed Description

This is a Standard C++ Library header.

## 7.290 `experimental/deque` File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_DEQUE`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::deque`

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (deque< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (deque< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 7.290.1 Detailed Description

This is a TS C++ Library header.

## 7.291 `expected` File Reference

### 7.291.1 Detailed Description

This is a Standard C++ Library header.

## 7.292 `lfts_config.h` File Reference

### 7.292.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 7.293 `ext/numeric_traits.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define _GLIBCXX_INT_N_TRAITS(T, WIDTH)`

## Typedefs

- `template<typename _Tp>`  
using `__gnu_cxx::__int_traits`

## Variables

- `template<typename _Value>`  
const int `__gnu_cxx::__numeric_traits_floating<_Value>::__digits10`
- `template<typename _Value>`  
const bool `__gnu_cxx::__numeric_traits_floating<_Value>::__is_signed`
- `template<typename _Value>`  
const int `__gnu_cxx::__numeric_traits_floating<_Value>::__max_digits10`
- `template<typename _Value>`  
const int `__gnu_cxx::__numeric_traits_floating<_Value>::__max_exponent10`
- `template<typename _Value>`  
const int `__gnu_cxx::__numeric_traits_integer<_Value>::__digits`
- `template<typename _Value>`  
const bool `__gnu_cxx::__numeric_traits_integer<_Value>::__is_signed`
- `template<typename _Value>`  
const \_Value `__gnu_cxx::__numeric_traits_integer<_Value>::__max`
- `template<typename _Value>`  
const \_Value `__gnu_cxx::__numeric_traits_integer<_Value>::__min`

### 7.293.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.294 propagate\_const File Reference

### Classes

- class `std::experimental::fundamentals_v2::propagate_const<_Tp>`

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_PROPAGATE_CONST`

### Typedefs

- `template<typename _Tp>`  
using `std::experimental::__propagate_const_elem_type`

### Functions

- `template<typename _Tp>`  
constexpr const \_Tp & `std::experimental::get_underlying` (const `propagate_const<_Tp>` &\_\_pt) noexcept
- `template<typename _Tp>`  
constexpr \_Tp & `std::experimental::get_underlying` (`propagate_const<_Tp>` &\_\_pt) noexcept

- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (const propagate\_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (nullptr_t, const propagate\_const< _Tp > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (const propagate\_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (nullptr_t, const propagate\_const< _Tp > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const _Tp &__t, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (const propagate\_const< _Tp > &__pt, const propagate\_const< _Up > &__pu)`
- `template<typename _Tp>`  
`constexpr enable\_if\_t< __is_swappable< _Tp >::value, void > std::experimental::swap (propagate\_const< _Tp > &__pt, propagate\_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

### 7.294.1 Detailed Description

This is a TS C++ Library header.

## 7.295 `simd` File Reference

### Macros

- `#define __cpp_lib_experimental_parallel_simd`
- `#define _GLIBCXX_EXPERIMENTAL_SIMD`

### 7.295.1 Detailed Description

This is a TS C++ Library header.

## 7.296 `aligned_buffer.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### 7.296.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.297 `atomicity.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_UNSIGNED_ATOMIC_WORD`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

### Functions

- `void __gnu_cxx::__atomic_add (volatile _Atomic_word * __mem, int __val)`
- `void __gnu_cxx::__atomic_add_dispatch (_Atomic_word * __mem, int __val)`
- `void __gnu_cxx::__atomic_add_single (_Atomic_word * __mem, int __val)`
- `_Atomic_word __gnu_cxx::__exchange_and_add (volatile _Atomic_word * __mem, int __val)`
- `_Atomic_word __gnu_cxx::__exchange_and_add_dispatch (_Atomic_word * __mem, int __val)`
- `_Atomic_word __gnu_cxx::__exchange_and_add_single (_Atomic_word * __mem, int __val)`
- `bool __gnu_cxx::__is_single_threaded () noexcept`

### 7.297.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.298 bitmap\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Bitmap\\_counter<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Ffit\\_finder<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::free\\_list](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)

### Macros

- [#define \\_BALLOC\\_ALIGN\\_BYTES](#)

### Enumerations

- enum { [bits\\_per\\_byte](#) , [bits\\_per\\_block](#) }

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (std::size\_t \*\_\_pmap, std::size\_t \_\_pos) throw ()
- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (std::size\_t \*\_\_pmap, std::size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare>  
\_ForwardIterator [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair>  
std::size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)
- std::size\_t [\\_\\_gnu\\_cxx::Bit\\_scan\\_forward](#) (std::size\_t \_\_num)
- template<typename \_Tp1, typename \_Tp2>  
bool [\\_\\_gnu\\_cxx::operator==](#) (const [bitmap\\_allocator](#)<\_Tp1> &, const [bitmap\\_allocator](#)<\_Tp2> &) throw ()

### Variables

- template<typename \_Tp>  
std::size\_t [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>::\\_S\\_block\\_size](#)
- template<typename \_Tp>  
[bitmap\\_allocator](#)<\_Tp>::\_BPVector::size\_type [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>::\\_S\\_last\\_dealloc\\_](#)  
[index](#)
- template<typename \_Tp>  
[bitmap\\_allocator](#)<\_Tp>::\_BPVector [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>::\\_S\\_mem\\_blocks](#)
- template<typename \_Tp>  
[bitmap\\_allocator](#)<\_Tp>::\_mutex\_type [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>::\\_S\\_mut](#)

#### 7.298.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.



## 7.298.2 Macro Definition Documentation

### **`_BALLOC_ALIGN_BYTES`**

```
#define _BALLOC_ALIGN_BYTES
```

The constant in the expression below is the alignment required in bytes.

## 7.299 cast.h File Reference

### Classes

- struct [`\_\_gnu\_cxx::\_Caster<\_ToType>`](#)

### Namespaces

- namespace [`\_\_gnu\_cxx`](#)

### Functions

- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_const\_pointer\_cast`](#) (`_FromType *``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_const\_pointer\_cast`](#) (const `_FromType &``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_dynamic\_pointer\_cast`](#) (`_FromType *``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_dynamic\_pointer\_cast`](#) (const `_FromType &``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast`](#) (`_FromType *``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast`](#) (const `_FromType &``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_static\_pointer\_cast`](#) (`_FromType *``__arg`)
- template<typename `_ToType`, typename `_FromType`>  
`_ToType` [`\_\_gnu\_cxx::\_\_static\_pointer\_cast`](#) (const `_FromType &``__arg`)

### 7.299.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

## 7.300 codecvt\_specializations.h File Reference

### Classes

- class [`std::codecvt<\_InternT, \_ExternT, encoding\_state>`](#)
- struct [`\_\_gnu\_cxx::encoding\_char\_traits<\_CharT>`](#)
- class [`\_\_gnu\_cxx::encoding\_state`](#)
- class [`std::encoding\_state`](#)

### Namespaces

- namespace [`\_\_gnu\_cxx`](#)
- namespace [`std`](#)

## Functions

- `template<typename _Tp>  
size_t std::__iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↵  
__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`

## Variables

- `template<typename _InternT, typename _ExternT>  
locale::id std::codecvt< _InternT, _ExternT, encoding_state >::id`

### 7.300.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.301 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum `_Lock_policy` { `_S_single` , `_S_mutex` , `_S_atomic` }

### Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

### Variables

- const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

### 7.301.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.302 `debug_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator< \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 7.302.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.303 enc\_filebuf.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.303.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.304 extptr\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::ExtPtr\\_allocator<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp>  
void \_\_gnu\_cxx::swap (\_ExtPtr\_allocator<\_Tp> &__larg, \_ExtPtr\_allocator<\_Tp> &__rarg)`

#### 7.304.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

#### Author

Bob Walters

An example allocator which uses an alternative pointer type from `bits/pointer.h`. Supports test cases which confirm container support for alternative pointers.

## 7.305 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.305.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.306 `mt_allocator.h` File Reference

### Classes

- struct `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >`
- class `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`
- class `__gnu_cxx::__mt_alloc_base< _Tp >`
- struct `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`
- class `__gnu_cxx::__pool< false >`
- class `__gnu_cxx::__pool< true >`
- struct `__gnu_cxx::__pool_base`

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define __thread_default`

### Typedefs

- typedef `void(* __gnu_cxx::__destroy_handler) (void *)`

### Functions

- template<typename `_Tp`, typename `_Poolp`>  
bool `__gnu_cxx::operator==` (const `__mt_alloc< _Tp, _Poolp >` &, const `__mt_alloc< _Tp, _Poolp >` &)

#### 7.306.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.307 `assoc_container.hpp` File Reference

### Classes

- class `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`
- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`
- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`
- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`
- class `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

**Macros**

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

**7.307.1 Detailed Description**

Contains associative containers.

**7.308 bin\_search\_tree.hpp File Reference****Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

**Variables**

- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>`  
`bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_allocator \_\_gnu\_pbds::`  
`detail::bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::s\_node\_allocator`

**7.308.1 Detailed Description**

Contains an implementation class for binary search tree.

**7.309 bin\_search\_tree\_/node\_iterators.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it< Node, Const\\_Iterator, Iterator, \\_Alloc >](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- `#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC`
- `#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC`

### 7.309.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 7.310 `ov_tree_map_/node_iterators.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC`
- `#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC`

### 7.310.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 7.311 `point_iterators.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

### 7.311.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 7.312 `bin_search_tree_/r_erase_fn_imps.hpp` File Reference

### 7.312.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 7.313 pat\_trie\_/r\_erase\_fn\_imps.hpp File Reference

### 7.313.1 Detailed Description

Contains an implementation class for pat\_trie.

## 7.314 bin\_search\_tree\_/rotate\_fn\_imps.hpp File Reference

### 7.314.1 Detailed Description

Contains imps for rotating nodes.

## 7.315 pat\_trie\_/rotate\_fn\_imps.hpp File Reference

### 7.315.1 Detailed Description

Contains imps for rotating nodes.

## 7.316 bin\_search\_tree\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.316.1 Detailed Description

Contains an implementation for bin\_search\_tree\_.

## 7.317 branch\_policy/traits.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

### 7.317.1 Detailed Description

Contains an implementation class for tree-like classes.

## 7.318 ov\_tree\_map\_/traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.318.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 7.319 `pat_trie_/traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, null\\_type, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.319.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 7.320 `rb_tree_map_/traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.320.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 7.321 `splay_tree_/traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.321.1 Detailed Description

Contains an implementation for `splay_tree_`.

## 7.322 `binary_heap_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)



## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

## Variables

- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
[binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::entry\_allocator [\\_\\_gnu\\_pbds::detail::binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::s\_entry\_allocator
- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
[binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::no\_throw\_copies\_t [\\_\\_gnu\\_pbds::detail::binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::s\_no\_throw\_copies\_ind
- `template<typename Value_Type, typename Cmp_Fn, typename _Alloc>`  
[binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::value\_allocator [\\_\\_gnu\\_pbds::detail::binary\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc >::s\_value\_allocator

### 7.322.1 Detailed Description

Contains an implementation class for a binary heap.

## 7.323 `binary_heap_/const_iterator.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#)< Value\_Type, Entry, Simple, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

### 7.323.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 7.324 `left_child_next_sibling_heap_/const_iterator.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_](#)< Node, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

**Macros**

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

**7.324.1 Detailed Description**

Contains an iterator class returned by the table's const find and insert methods.

**7.325 unordered\_iterator/const\_iterator.hpp File Reference****7.325.1 Detailed Description**

Contains an iterator class used for const ranging over the elements of the table.

This file is intended to be included inside a class definition, with `PB_DS_CLASS_C_DEC` defined to the name of the enclosing class.

**7.326 bin\_search\_tree\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.326.1 Detailed Description**

Contains an implementation class for `bin_search_tree_`.

**7.327 binary\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.327.1 Detailed Description**

Contains an implementation class for `binary_heap_`.

**7.328 binomial\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.328.1 Detailed Description**

Contains an implementation for `binomial_heap_`.

**7.329 binomial\_heap\_base\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.329.1 Detailed Description**

Contains an implementation class for a base of binomial heaps.

**7.330 left\_child\_next\_sibling\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.330.1 Detailed Description**

Contains an implementation class for `left_child_next_sibling_heap_`.

**7.331 ov\_tree\_map\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.331.1 Detailed Description**

Contains an implementation class for `ov_tree_`.

**7.332 pairing\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference****7.332.1 Detailed Description**

Contains an implementation class for a pairing heap.

### 7.333 pat\_trie\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 7.333.1 Detailed Description

Contains an implementation class for pat\_trie.

### 7.334 rb\_tree\_map\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 7.334.1 Detailed Description

Contains an implementation for rb\_tree\_.

### 7.335 rc\_binomial\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 7.335.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

### 7.336 splay\_tree\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 7.336.1 Detailed Description

Contains an implementation class for splay\_tree\_.

### 7.337 thin\_heap\_/constructors\_destructor\_fn\_imps.hpp File Reference

#### 7.337.1 Detailed Description

Contains an implementation for thin\_heap\_.

### 7.338 bin\_search\_tree\_/debug\_fn\_imps.hpp File Reference

#### 7.338.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

### 7.339 binary\_heap\_/debug\_fn\_imps.hpp File Reference

#### 7.339.1 Detailed Description

Contains an implementation class for a binary\_heap.

### 7.340 binomial\_heap\_/debug\_fn\_imps.hpp File Reference

#### 7.340.1 Detailed Description

Contains an implementation for binomial\_heap\_.

### 7.341 binomial\_heap\_base\_/debug\_fn\_imps.hpp File Reference

#### 7.341.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 7.342 cc\_hash\_table\_map\_/debug\_fn\_imps.hpp File Reference

#### 7.342.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

### **7.343 gp\_hash\_table\_map\_/debug\_fn\_imps.hpp File Reference**

#### **7.343.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

### **7.344 left\_child\_next\_sibling\_heap\_/debug\_fn\_imps.hpp File Reference**

#### **7.344.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

### **7.345 list\_update\_map\_/debug\_fn\_imps.hpp File Reference**

#### **7.345.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

### **7.346 ov\_tree\_map\_/debug\_fn\_imps.hpp File Reference**

#### **7.346.1 Detailed Description**

Contains an implementation class for ov\_tree\_.

### **7.347 pairing\_heap\_/debug\_fn\_imps.hpp File Reference**

#### **7.347.1 Detailed Description**

Contains an implementation class for a pairing heap.

### **7.348 pat\_trie\_/debug\_fn\_imps.hpp File Reference**

#### **7.348.1 Detailed Description**

Contains an implementation class for pat\_trie\_.

### **7.349 rb\_tree\_map\_/debug\_fn\_imps.hpp File Reference**

#### **7.349.1 Detailed Description**

Contains an implementation for rb\_tree\_.

### **7.350 rc\_binomial\_heap\_/debug\_fn\_imps.hpp File Reference**

#### **7.350.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

### **7.351 splay\_tree\_/debug\_fn\_imps.hpp File Reference**

#### **7.351.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

### **7.352 thin\_heap\_/debug\_fn\_imps.hpp File Reference**

#### **7.352.1 Detailed Description**

Contains an implementation for thin\_heap\_.

## 7.353 entry\_cmp.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp< \\_VTp, Cmp\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp< \\_VTp, Cmp\\_Fn, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp< \\_VTp, Cmp\\_Fn, \\_Alloc, false >::type](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.353.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 7.354 entry\_pred.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred< \\_VTp, Pred, \\_Alloc, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.354.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 7.355 bin\_search\_tree\_/erase\_fn\_imps.hpp File Reference

### 7.355.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 7.356 binary\_heap\_/erase\_fn\_imps.hpp File Reference

### 7.356.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 7.357 binomial\_heap\_base\_/erase\_fn\_imps.hpp File Reference

### 7.357.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 7.358 cc\_hash\_table\_map\_/erase\_fn\_imps.hpp File Reference

### 7.358.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s erase related functions.

**7.359 gp\_hash\_table\_map\_/erase\_fn\_imps.hpp File Reference****7.359.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s erase related functions.

**7.360 left\_child\_next\_sibling\_heap\_/erase\_fn\_imps.hpp File Reference****7.360.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

**7.361 list\_update\_map\_/erase\_fn\_imps.hpp File Reference****7.361.1 Detailed Description**

Contains implementations of lu\_map\_.

**7.362 ov\_tree\_map\_/erase\_fn\_imps.hpp File Reference****7.362.1 Detailed Description**

Contains an implementation class for ov\_tree\_.

**7.363 pairing\_heap\_/erase\_fn\_imps.hpp File Reference****7.363.1 Detailed Description**

Contains an implementation class for a pairing heap.

**7.364 pat\_trie\_/erase\_fn\_imps.hpp File Reference****7.364.1 Detailed Description**

Contains an implementation class for pat\_trie.

**7.365 rb\_tree\_map\_/erase\_fn\_imps.hpp File Reference****7.365.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**7.366 rc\_binomial\_heap\_/erase\_fn\_imps.hpp File Reference****7.366.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

**7.367 splay\_tree\_/erase\_fn\_imps.hpp File Reference****7.367.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

**7.368 thin\_heap\_/erase\_fn\_imps.hpp File Reference****7.368.1 Detailed Description**

Contains an implementation for thin\_heap\_.

## 7.369 bin\_search\_tree\_/find\_fn\_imps.hpp File Reference

### 7.369.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 7.370 binary\_heap\_/find\_fn\_imps.hpp File Reference

### 7.370.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 7.371 binomial\_heap\_base\_/find\_fn\_imps.hpp File Reference

### 7.371.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 7.372 cc\_hash\_table\_map\_/find\_fn\_imps.hpp File Reference

### 7.372.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s find related functions.

## 7.373 gp\_hash\_table\_map\_/find\_fn\_imps.hpp File Reference

### 7.373.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s find related functions.

## 7.374 list\_update\_map\_/find\_fn\_imps.hpp File Reference

### 7.374.1 Detailed Description

Contains implementations of lu\_map\_.

## 7.375 pairing\_heap\_/find\_fn\_imps.hpp File Reference

### 7.375.1 Detailed Description

Contains an implementation class for a pairing heap.

## 7.376 pat\_trie\_/find\_fn\_imps.hpp File Reference

### 7.376.1 Detailed Description

Contains an implementation class for pat\_trie.

## 7.377 rb\_tree\_map\_/find\_fn\_imps.hpp File Reference

### 7.377.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 7.378 splay\_tree\_/find\_fn\_imps.hpp File Reference

### 7.378.1 Detailed Description

Contains an implementation class for splay\_tree\_.

### **7.379 thin\_heap\_/find\_fn\_imps.hpp File Reference**

#### **7.379.1 Detailed Description**

Contains an implementation for thin\_heap\_.

### **7.380 bin\_search\_tree\_/info\_fn\_imps.hpp File Reference**

#### **7.380.1 Detailed Description**

Contains an implementation class for bin\_search\_tree\_.

### **7.381 binary\_heap\_/info\_fn\_imps.hpp File Reference**

#### **7.381.1 Detailed Description**

Contains an implementation class for a binary\_heap.

### **7.382 cc\_hash\_table\_map\_/info\_fn\_imps.hpp File Reference**

#### **7.382.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s entire container info related functions.

### **7.383 gp\_hash\_table\_map\_/info\_fn\_imps.hpp File Reference**

#### **7.383.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s entire container info related functions.

### **7.384 left\_child\_next\_sibling\_heap\_/info\_fn\_imps.hpp File Reference**

#### **7.384.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

### **7.385 list\_update\_map\_/info\_fn\_imps.hpp File Reference**

#### **7.385.1 Detailed Description**

Contains implementations of lu\_map\_.

### **7.386 ov\_tree\_map\_/info\_fn\_imps.hpp File Reference**

#### **7.386.1 Detailed Description**

Contains an implementation class for ov\_tree\_.

### **7.387 pat\_trie\_/info\_fn\_imps.hpp File Reference**

#### **7.387.1 Detailed Description**

Contains an implementation class for pat\_trie.

### **7.388 rb\_tree\_map\_/info\_fn\_imps.hpp File Reference**

#### **7.388.1 Detailed Description**

Contains an implementation for rb\_tree\_.



## 7.389 `splay_tree_/info_fn_imps.hpp` File Reference

### 7.389.1 Detailed Description

Contains an implementation.

## 7.390 `bin_search_tree_/insert_fn_imps.hpp` File Reference

### 7.390.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 7.391 `binary_heap_/insert_fn_imps.hpp` File Reference

### 7.391.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 7.392 `binomial_heap_base_/insert_fn_imps.hpp` File Reference

### 7.392.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 7.393 `cc_hash_table_map_/insert_fn_imps.hpp` File Reference

### 7.393.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions.

## 7.394 `gp_hash_table_map_/insert_fn_imps.hpp` File Reference

### 7.394.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions.

## 7.395 `left_child_next_sibling_heap_/insert_fn_imps.hpp` File Reference

### 7.395.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 7.396 `list_update_map_/insert_fn_imps.hpp` File Reference

### 7.396.1 Detailed Description

Contains implementations of `lu_map_`.

## 7.397 `ov_tree_map_/insert_fn_imps.hpp` File Reference

### 7.397.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 7.398 `pairing_heap_/insert_fn_imps.hpp` File Reference

### 7.398.1 Detailed Description

Contains an implementation class for a pairing heap.

**7.399 rb\_tree\_map\_/insert\_fn\_imps.hpp File Reference****7.399.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**7.400 rc\_binomial\_heap\_/insert\_fn\_imps.hpp File Reference****7.400.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

**7.401 splay\_tree\_/insert\_fn\_imps.hpp File Reference****7.401.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

**7.402 thin\_heap\_/insert\_fn\_imps.hpp File Reference****7.402.1 Detailed Description**

Contains an implementation for thin\_heap\_.

**7.403 bin\_search\_tree\_/iterators\_fn\_imps.hpp File Reference****7.403.1 Detailed Description**

Contains an implementation class for bin\_search\_tree\_.

**7.404 binary\_heap\_/iterators\_fn\_imps.hpp File Reference****7.404.1 Detailed Description**

Contains an implementation class for a binary\_heap.

**7.405 cc\_hash\_table\_map\_/iterators\_fn\_imps.hpp File Reference****7.405.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s iterators related functions, e.g., begin().

**7.406 left\_child\_next\_sibling\_heap\_/iterators\_fn\_imps.hpp File Reference****7.406.1 Detailed Description**

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

**7.407 list\_update\_map\_/iterators\_fn\_imps.hpp File Reference****7.407.1 Detailed Description**

Contains implementations of lu\_map\_.

**7.408 ov\_tree\_map\_/iterators\_fn\_imps.hpp File Reference****7.408.1 Detailed Description**

Contains an implementation class for ov\_tree\_.

## 7.409 pat\_trie\_/iterators\_fn\_imps.hpp File Reference

### 7.409.1 Detailed Description

Contains an implementation class for pat\_trie.

## 7.410 binary\_heap\_/point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_ < Value\\_Type, Entry, Simple, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.410.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 7.411 left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_ < Node, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 7.411.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 7.412 unordered\_iterator/point\_const\_iterator.hpp File Reference

### 7.412.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

- This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 7.413 bin\_search\_tree\_/policy\_access\_fn\_imps.hpp File Reference

### 7.413.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 7.414 binary\_heap\_/policy\_access\_fn\_imps.hpp File Reference

### 7.414.1 Detailed Description

Contains an implementation class for a binary\_heap.

### 7.415 `cc_hash_table_map_/policy_access_fn_imps.hpp` File Reference

#### 7.415.1 Detailed Description

Contains implementations of `cc_ht_map_`'s policy access functions.

### 7.416 `gp_hash_table_map_/policy_access_fn_imps.hpp` File Reference

#### 7.416.1 Detailed Description

Contains implementations of `gp_ht_map_`'s policy access functions.

### 7.417 `left_child_next_sibling_heap_/policy_access_fn_imps.hpp` File Reference

#### 7.417.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

### 7.418 `ov_tree_map_/policy_access_fn_imps.hpp` File Reference

#### 7.418.1 Detailed Description

Contains an implementation class for `ov_tree`.

### 7.419 `pat_trie_/policy_access_fn_imps.hpp` File Reference

#### 7.419.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 7.420 `resize_policy.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::resize\\_policy<\\_Tp>](#)

#### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### Variables

- `template<typename _Tp>`  
`const _Tp __gnu_pbds::detail::resize_policy<_Tp>::min_size`

#### 7.420.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 7.421 `bin_search_tree_/split_join_fn_imps.hpp` File Reference

#### 7.421.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 7.422 `binary_heap_/split_join_fn_imps.hpp` File Reference

### 7.422.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 7.423 `binomial_heap_base_/split_join_fn_imps.hpp` File Reference

### 7.423.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 7.424 `ov_tree_map_/split_join_fn_imps.hpp` File Reference

### 7.424.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 7.425 `pairing_heap_/split_join_fn_imps.hpp` File Reference

### 7.425.1 Detailed Description

Contains an implementation class for a pairing heap.

## 7.426 `rb_tree_map_/split_join_fn_imps.hpp` File Reference

### 7.426.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 7.427 `rc_binomial_heap_/split_join_fn_imps.hpp` File Reference

### 7.427.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 7.428 `splay_tree_/split_join_fn_imps.hpp` File Reference

### 7.428.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 7.429 `thin_heap_/split_join_fn_imps.hpp` File Reference

### 7.429.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 7.430 `binary_heap_/trace_fn_imps.hpp` File Reference

### 7.430.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 7.431 `cc_hash_table_map_/trace_fn_imps.hpp` File Reference

### 7.431.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

## 7.432 `gp_hash_table_map_/trace_fn_imps.hpp` File Reference

### 7.432.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

## 7.433 `left_child_next_sibling_heap_/trace_fn_imps.hpp` File Reference

### 7.433.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 7.434 `list_update_map_/trace_fn_imps.hpp` File Reference

### 7.434.1 Detailed Description

Contains implementations of `lu_map_`.

## 7.435 `pat_trie_/trace_fn_imps.hpp` File Reference

### 7.435.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 7.436 `rc_binomial_heap_/trace_fn_imps.hpp` File Reference

### 7.436.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 7.437 `thin_heap_/trace_fn_imps.hpp` File Reference

### 7.437.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 7.438 `binomial_heap_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 7.438.1 Detailed Description

Contains an implementation class for a binomial heap.

## 7.439 binomial\_heap\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_ASSERT\_BASE\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_ASSERT\_VALID\_COND**(X, \_StrictlyBinomial)
- #define **PB\_DS\_B\_HEAP\_BASE**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**

#### 7.439.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 7.440 branch\_policy.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Itr, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Cltr, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.440.1 Detailed Description

Contains a base class for branch policies.

## 7.441 null\_node\_metadata.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator< Key, Data, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.441.1 Detailed Description

Contains an implementation class for tree-like classes.

## 7.442 cc\_ht\_map.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

## Variables

- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_↵_Hash_Fn, typename Resize_Policy>`  
`cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::const_↵_iterator` `\_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵_Hash_Fn, Resize_Policy >::s_const_end_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_↵_Hash_Fn, typename Resize_Policy>`  
`cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::iterator` `\_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_end_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_↵_Hash_Fn, typename Resize_Policy>`  
`cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry_↵_allocator` `\_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵_Hash_Fn, Resize_Policy >::s_entry_allocator`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_↵_Hash_Fn, typename Resize_Policy>`  
`cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry_↵_pointer_allocator` `\_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_entry_pointer_allocator`

### 7.442.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

## 7.443 `cmp_fn_imps.hpp` File Reference

### 7.443.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container comparison related functions.

## 7.444 `cond_key_dtor_entry_dealtor.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor](#)< Entry, \_Alloc >

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)



**7.444.1 Detailed Description**

Contains a conditional key destructor, used for exception handling.

**7.445 cc\_hash\_table\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference****7.445.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

**7.446 gp\_hash\_table\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference****7.446.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

**7.447 list\_update\_map\_/constructor\_destructor\_fn\_imps.hpp File Reference****7.448 cc\_hash\_table\_map\_/constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference****7.448.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

**7.449 gp\_hash\_table\_map\_/constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference****7.449.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

**7.450 cc\_hash\_table\_map\_/constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference****7.450.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

**7.451 gp\_hash\_table\_map\_/constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference****7.451.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

**7.452 cc\_hash\_table\_map\_/debug\_no\_store\_hash\_fn\_imps.hpp File Reference****7.452.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

**7.453 gp\_hash\_table\_map\_/debug\_no\_store\_hash\_fn\_imps.hpp File Reference****7.453.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

**7.454 cc\_hash\_table\_map\_/debug\_store\_hash\_fn\_imps.hpp File Reference****7.454.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s debug-mode functions.

**7.455 gp\_hash\_table\_map\_/debug\_store\_hash\_fn\_imps.hpp File Reference****7.455.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s debug-mode functions.

**7.456 entry\_list\_fn\_imps.hpp File Reference****7.456.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s entry-list related functions.

**7.457 cc\_hash\_table\_map\_/erase\_no\_store\_hash\_fn\_imps.hpp File Reference****7.457.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s erase related functions, when the hash value is not stored.

**7.458 gp\_hash\_table\_map\_/erase\_no\_store\_hash\_fn\_imps.hpp File Reference****7.458.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is not stored.

**7.459 cc\_hash\_table\_map\_/erase\_store\_hash\_fn\_imps.hpp File Reference****7.459.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s erase related functions, when the hash value is stored.

**7.460 gp\_hash\_table\_map\_/erase\_store\_hash\_fn\_imps.hpp File Reference****7.460.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is stored.

**7.461 cc\_hash\_table\_map\_/find\_store\_hash\_fn\_imps.hpp File Reference****7.461.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s find related functions, when the hash value is stored.

**7.462 gp\_hash\_table\_map\_/find\_store\_hash\_fn\_imps.hpp File Reference****7.462.1 Detailed Description**

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is stored.

**7.463 cc\_hash\_table\_map\_/insert\_no\_store\_hash\_fn\_imps.hpp File Reference****7.463.1 Detailed Description**

Contains implementations of cc\_ht\_map\_'s insert related functions, when the hash value is not stored.

**7.464 `gp_hash_table_map/_insert_no_store_hash_fn_imps.hpp` File Reference****7.464.1 Detailed Description**

Contains implementations of `gp_ht_map_'s` insert related functions, when the hash value is not stored.

**7.465 `cc_hash_table_map/_insert_store_hash_fn_imps.hpp` File Reference****7.465.1 Detailed Description**

Contains implementations of `cc_ht_map_'s` insert related functions, when the hash value is stored.

**7.466 `gp_hash_table_map/_insert_store_hash_fn_imps.hpp` File Reference****7.466.1 Detailed Description**

Contains implementations of `gp_ht_map_'s` find related functions, when the hash value is stored.

**7.467 `cc_hash_table_map/_resize_fn_imps.hpp` File Reference****7.467.1 Detailed Description**

Contains implementations of `cc_ht_map_'s` resize related functions.

**7.468 `gp_hash_table_map/_resize_fn_imps.hpp` File Reference****7.468.1 Detailed Description**

Contains implementations of `gp_ht_map_'s` resize related functions.

**7.469 `cc_hash_table_map/_resize_no_store_hash_fn_imps.hpp` File Reference****7.469.1 Detailed Description**

Contains implementations of `cc_ht_map_'s` resize related functions, when the hash value is not stored.

**7.470 `gp_hash_table_map/_resize_no_store_hash_fn_imps.hpp` File Reference****7.470.1 Detailed Description**

Contains implementations of `gp_ht_map_'s` resize related functions, when the hash value is not stored.

**7.471 `cc_hash_table_map/_resize_store_hash_fn_imps.hpp` File Reference****7.471.1 Detailed Description**

Contains implementations of `cc_ht_map_'s` resize related functions, when the hash value is stored.

**7.472 `gp_hash_table_map/_resize_store_hash_fn_imps.hpp` File Reference****7.472.1 Detailed Description**

Contains implementations of `gp_ht_map_'s` resize related functions, when the hash value is stored.

**7.473 `size_fn_imps.hpp` File Reference****7.473.1 Detailed Description**

Contains implementations of `cc_ht_map_'s` entire container size related functions.

## 7.474 cond\_dealtor.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Variables

- template<typename Entry, class \_Alloc>  
[cond\\_dealtor< Entry, \\_Alloc >::entry\\_allocator](#) [\\_\\_gnu\\_pbds::detail::cond\\_dealtor< Entry, \\_Alloc >::s\\_alloc](#)

#### 7.474.1 Detailed Description

Contains a conditional deallocator.

## 7.475 container\_base\_dispatch.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 7.475.1 Detailed Description

Contains associative container dispatching.

## 7.476 debug\_map\_base.hpp File Reference

### 7.476.1 Detailed Description

Contains a debug-mode base for all maps.

## 7.477 eq\_by\_less.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less< Key, Cmp\\_Fn >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.477.1 Detailed Description

Contains an equivalence function.

## 7.478 hash\_eq\_fn.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.478.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

## 7.479 find\_no\_store\_hash\_fn\_imps.hpp File Reference

### 7.479.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s find related functions, when the hash value is not stored.

## 7.480 gp\_ht\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::gp\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Probe\\_Fn, Probe\\_Fn, Resize\\_Policy >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

## Variables

- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↔_Probe_Fn, typename Probe_Fn, typename Resize_Policy>`  
`gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy`  
`>::const_iterator __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,`  
`Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_const_end_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↔_Probe_Fn, typename Probe_Fn, typename Resize_Policy>`  
`gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy`  
`>::iterator __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↔_Probe_Fn, Probe_Fn, Resize_Policy >::s_end_it`
- `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↔_Probe_Fn, typename Probe_Fn, typename Resize_Policy>`  
`gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy`  
`>::entry_allocator __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,`  
`Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_entry_allocator`

### 7.480.1 Detailed Description

Contains an implementation class for general probing hash.

## 7.481 iterator\_fn\_imps.hpp File Reference

### 7.481.1 Detailed Description

Contains implementations of `gp_ht_map`'s iterators related functions, e.g., `begin()`.

## 7.482 direct\_mask\_range\_hashing\_imp.hpp File Reference

### 7.482.1 Detailed Description

Contains a range-hashing policy implementation

## 7.483 direct\_mod\_range\_hashing\_imp.hpp File Reference

### 7.483.1 Detailed Description

Contains a range-hashing policy implementation

## 7.484 linear\_probe\_fn\_imp.hpp File Reference

### 7.484.1 Detailed Description

Contains a probe policy implementation

## 7.485 mask\_based\_range\_hashing.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Variables

- template<typename Size\_Type>  
const [mask\\_based\\_range\\_hashing< Size\\_Type >::size\\_type](#) [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >::s\\_highest\\_bit\\_1](#)
- template<typename Size\_Type>  
const [mask\\_based\\_range\\_hashing< Size\\_Type >::size\\_type](#) [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing< Size\\_Type >::s\\_num\\_bits\\_in\\_size\\_type](#)

#### 7.485.1 Detailed Description

Contains a range hashing policy base.

## 7.486 mod\_based\_range\_hashing.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.486.1 Detailed Description

Contains a range hashing policy base.

## 7.487 probe\_fn\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.487.1 Detailed Description

Contains a probe policy base.

## 7.488 quadratic\_probe\_fn\_imp.hpp File Reference

#### 7.488.1 Detailed Description

Contains a probe policy implementation

## 7.489 ranged\_hash\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 7.489.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

## 7.490 ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 7.490.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probing.

## 7.491 sample\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_probe\\_fn](#)



**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**7.491.1 Detailed Description**

Contains a sample probe policy.

**7.492 sample\_range\_hashing.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_range\\_hashing](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**7.492.1 Detailed Description**

Contains a range hashing policy.

**7.493 sample\_ranged\_hash\_fn.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_hash\\_fn](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**7.493.1 Detailed Description**

Contains a ranged hash policy.

**7.494 sample\_ranged\_probe\_fn.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_probe\\_fn](#)

**Namespaces**

- namespace [\\_\\_gnu\\_pbds](#)

**7.494.1 Detailed Description**

Contains a ranged probe policy.

**7.495 left\_child\_next\_sibling\_heap\_.hpp File Reference****Classes**

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap< Value\\_Type, Cmp\\_Fn, Node\\_Metadata, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## Variables

- `template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>  
left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::no_throw_copies_t \_\_gnu\_↵  
\_pbds::detail::left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_no_↵  
throw_copies_ind`
- `template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>  
left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::node_allocator \_\_gnu\_↵  
pbds::detail::left\_child\_next\_sibling\_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_node_↵  
allocator`

### 7.495.1 Detailed Description

Contains an implementation class for a basic heap.

## 7.496 [left\\_child\\_next\\_sibling\\_heap\\_/node.hpp](#) File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_< \\_Value, \\_Metadata, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.496.1 Detailed Description

Contains an implementation struct for this type of heap's node.

## 7.497 [rb\\_tree\\_map\\_/node.hpp](#) File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.497.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 7.498 [splay\\_tree\\_/node.hpp](#) File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.498.1 Detailed Description

Contains an implementation struct for splay\_tree\_'s node.

## 7.499 entry\_metadata\_base.hpp File Reference

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.499.1 Detailed Description

Contains an implementation for a list update map.

## 7.500 lu\_map\_.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

## Variables

- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>  
lu\_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::entry_allocator \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_entry\_allocator`
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>  
Eq_Fn \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_eq\_fn`
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>  
type_to_type< typename lu\_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::update_metadata > \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_metadata\_type\_indicator`
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>  
null\_type \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_null\_type`
- `template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>  
Update_Policy \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >::s\_update\_  
policy`

### 7.500.1 Detailed Description

Contains a list update map.

## 7.501 lu\_counter\_metadata.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.501.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

## 7.502 sample\_update\_policy.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_update\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

#### 7.502.1 Detailed Description

Contains a sample policy for list update containers.

## 7.503 ov\_tree\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >::cond\\_dtor< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

### Variables

- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>`  
`ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::metadata_allocator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_metadata_alloc`
- `template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>`  
`ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::value_allocator \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_value_alloc`

### 7.503.1 Detailed Description

Contains an implementation class for ov\_tree.

## 7.504 pairing\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pairing\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_ASSERT\_NODE\_CONSISTENT**(\_Node, \_Bool)
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_P\_HEAP\_BASE**

### 7.504.1 Detailed Description

Contains an implementation class for a pairing heap.

## 7.505 insert\_join\_fn\_imps.hpp File Reference

### 7.505.1 Detailed Description

Contains an implementation class for pat\_trie.

## 7.506 pat\_trie\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_ASSERT\_NODE\_VALID**(X)
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_PAT\_TRIE\_NAME**
- #define **PB\_DS\_PAT\_TRIE\_TRAITS\_BASE**
- #define **PB\_DS\_RECURSIVE\_COUNT\_LEAFS**(X)

## Variables

- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::head\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_head\_allocator
- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::inode\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_inode\_allocator
- `template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>`  
[pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::leaf\_allocator [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map](#)< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >::s\_leaf\_allocator

### 7.506.1 Detailed Description

Contains an implementation class for a patricia tree.

## 7.507 pat\_trie\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter](#)< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head](#)< \_ATraits, Metadata >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter](#)< Node, Leaf, Head, Inode, Is\_Forward\_Iterator >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf](#)< \_ATraits, Metadata >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata](#)< Metadata, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata](#)< null\_type, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base](#)< \_ATraits, Metadata >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer](#)< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter](#)< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >::const\_iterator
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode](#)< \_ATraits, Metadata >::iterator
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define` [PB\\_DS\\_CLASS\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_CLASS\\_T\\_DEC](#)
- `#define` [PB\\_DS\\_CONST\\_IT\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_CONST\\_ODIR\\_IT\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_IT\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_ODIR\\_IT\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_PAT\\_TRIE\\_NODE\\_CONST\\_ITERATOR\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_PAT\\_TRIE\\_NODE\\_ITERATOR\\_C\\_DEC](#)

### 7.507.1 Detailed Description

Contains the base class for a patricia tree.

## 7.508 split\_fn\_imps.hpp File Reference

### 7.508.1 Detailed Description

Contains an implementation class for pat\_trie.

## 7.509 synth\_access\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_SYNTH\_E\_ACCESS\_TRAITS\_C\_DEC**
- #define **PB\_DS\_SYNTH\_E\_ACCESS\_TRAITS\_T\_DEC**

### Variables

- template<typename Type\_Traits, bool Set, typename \_ATraits>  
integral\_constant< int, Set > [\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >::s\\_](#)↔  
[set\\_ind](#)

### 7.509.1 Detailed Description

Contains an implementation class for a patricia tree.

## 7.510 update\_fn\_imps.hpp File Reference

### 7.510.1 Detailed Description

Contains an implementation class for pat\_trie\_.

## 7.511 priority\_queue\_base\_dispatch.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< \\_VTp, Cmp\\_Fn, \\_Alloc, binary\\_heap\\_tag, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< \\_VTp, Cmp\\_Fn, \\_Alloc, binomial\\_heap\\_tag, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< \\_VTp, Cmp\\_Fn, \\_Alloc, pairing\\_heap\\_tag, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< \\_VTp, Cmp\\_Fn, \\_Alloc, rc\\_binomial\\_heap\\_tag, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< \\_VTp, Cmp\\_Fn, \\_Alloc, thin\\_heap\\_tag, null\\_type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- #define **PB\_DS\_ASSERT\_VALID(X)**
- #define **PB\_DS\_DEBUG\_VERIFY(\_Cond)**

### 7.511.1 Detailed Description

Contains an pqiative container dispatching base.

## 7.512 `rb_tree_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

### 7.512.1 Detailed Description

Contains an implementation for Red Black trees.

## 7.513 `rc.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.513.1 Detailed Description

Contains a redundant (binary counter).

## 7.514 `rc_binomial_heap_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`



### 7.514.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

## 7.515 cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp File Reference

### 7.515.1 Detailed Description

Contains a resize trigger implementation.

## 7.516 hash\_exponential\_size\_policy\_imp.hpp File Reference

### 7.516.1 Detailed Description

Contains a resize size policy implementation.

## 7.517 hash\_load\_check\_resize\_trigger\_imp.hpp File Reference

### 7.517.1 Detailed Description

Contains a resize trigger implementation.

## 7.518 hash\_load\_check\_resize\_trigger\_size\_base.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.518.1 Detailed Description

Contains an base holding size for some resize policies.

## 7.519 hash\_prime\_size\_policy\_imp.hpp File Reference

### 7.519.1 Detailed Description

Contains a resize size policy implementation.

## 7.520 hash\_standard\_resize\_policy\_imp.hpp File Reference

### 7.520.1 Detailed Description

Contains a resize policy implementation.

## 7.521 sample\_resize\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.521.1 Detailed Description

Contains a sample resize policy for hash tables.

## 7.522 `sample_resize_trigger.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_trigger](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.522.1 Detailed Description

Contains a sample resize trigger policy class.

## 7.523 `sample_size_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_size\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.523.1 Detailed Description

Contains a sample size resize-policy.

## 7.524 `splay_fn_imps.hpp` File Reference

### 7.524.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 7.525 `splay_tree_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

### 7.525.1 Detailed Description

Contains an implementation class for splay trees.

## 7.526 `standard_policies.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::default\\_comb\\_hash\\_fn](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_eq\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_hash\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_probe\\_fn< Comb\\_Probe\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_resize\\_policy< Comb\\_Hash\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< std::basic\\_string< Char, Char\\_Traits, std::allocator< char > > >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_update\\_policy](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

### Enumerations

- enum { `default_store_hash` }

### 7.526.1 Detailed Description

Contains standard policies for containers.

### 7.526.2 Enumeration Type Documentation

#### anonymous enum

`anonymous enum`

Enumeration for default behavior of stored hash data.

## 7.527 `thin_heap_.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## Enumerations

- `enum { num_distinct_rank_bounds }`

## Variables

- `static const std::size_t __gnu_pbds::detail::g_a_rank_bounds [num_distinct_rank_bounds]`

### 7.527.1 Detailed Description

Contains an implementation class for a thin heap.

## 7.528 tree\_policy/node\_metadata\_selector.hpp File Reference

### Classes

- `struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- `struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- `struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### 7.528.1 Detailed Description

Contains an implementation class for trees.

## 7.529 trie\_policy/node\_metadata\_selector.hpp File Reference

### Classes

- `struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
- `struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- `struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`

### Namespaces

- namespace `__gnu_pbds`

### 7.529.1 Detailed Description

Contains an implementation class for tries.

## 7.530 tree\_policy/order\_statistics\_imp.hpp File Reference

### 7.530.1 Detailed Description

Contains forward declarations for `order_statistics_key`

## 7.531 trie\_policy/order\_statistics\_imp.hpp File Reference

### 7.531.1 Detailed Description

Contains forward declarations for order\_statistics\_key

## 7.532 sample\_tree\_node\_update.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update](#)< Const\_Node\_Iter, Node\_Iter, Cmp\_Fn, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.532.1 Detailed Description

Contains a sample node update functor.

## 7.533 tree\_trace\_base.hpp File Reference

### 7.533.1 Detailed Description

Contains tree-related policies.

## 7.534 prefix\_search\_node\_update\_imp.hpp File Reference

### 7.534.1 Detailed Description

Contains an implementation of prefix\_search\_node\_update.

## 7.535 sample\_trie\_access\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.535.1 Detailed Description

Contains a sample probe policy.

## 7.536 sample\_trie\_node\_update.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update](#)< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.536.1 Detailed Description

Contains a sample node update functor.

## 7.537 `trie_policy_base.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base< Node\\_Cltr, Node\\_Itr, \\_ATraits, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 7.537.1 Detailed Description

Contains an implementation of `trie_policy_base`.

## 7.538 `trie_string_access_traits_imp.hpp` File Reference

### 7.538.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

## 7.539 `type_utils.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

### Typedefs

- typedef `std::tr1::integral_constant< int, 0 >` [\\_\\_gnu\\_pbds::detail::false\\_type](#)
- typedef `std::tr1::integral_constant< int, 1 >` [\\_\\_gnu\\_pbds::detail::true\\_type](#)

### 7.539.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

## 7.540 `types_traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, Mapped, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, null\\_type, \\_Alloc, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, Mapped >](#)
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies< Key, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::rebind\\_traits< \\_Alloc, T >](#)

- struct [\\_\\_gnu\\_pbds::detail::select\\_value\\_type< Key, Mapped >](#)
- struct [\\_\\_gnu\\_pbds::detail::select\\_value\\_type< Key, null\\_type >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th, Store\\_Hash >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >](#)
- struct [\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >](#)
- struct [\\_\\_gnu\\_pbds::detail::types\\_traits< Key, Mapped, \\_Alloc, Store\\_Hash >](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Variables

- template<typename Key, typename \_Alloc, bool Store\_Hash>  
[null\\_type](#) [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type< Key, null\\_type, \\_Alloc, Store\\_Hash >::s\\_null\\_type](#)

### 7.540.1 Detailed Description

Contains a traits class of types used by containers.

## 7.541 iterator.hpp File Reference

### 7.541.1 Detailed Description

Contains an iterator\_ class used for ranging over the elements of the table.

This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 7.542 point\_iterator.hpp File Reference

### 7.542.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 7.543 exception.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error\(\)](#)

### 7.543.1 Detailed Description

Contains exception classes.

## 7.544 hash\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

### Enumerations

- enum { `num_distinct_sizes_16_bit` , `num_distinct_sizes_32_bit` , `num_distinct_sizes_64_bit` , `num_↵distinct_sizes` }

### Variables

- static const std::size\_t `__gnu_pbds::detail::g_a_sizes` [num\_distinct\_sizes]



### 7.544.1 Detailed Description

Contains hash-related policies.

## 7.545 list\_update\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::lu\\_counter\\_policy< Max\\_Count, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::lu\\_move\\_to\\_front\\_policy< \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.545.1 Detailed Description

Contains policies for list update containers.

## 7.546 priority\_queue.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::priority\\_queue< \\_Tv, Cmp\\_Fn, Tag, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 7.546.1 Detailed Description

Contains priority\_queues.

## 7.547 tag\_and\_trait.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_tag](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#)

- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#)
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- struct [\\_\\_gnu\\_pbds::null\\_node\\_update< \\_Tp1, \\_Tp2, \\_Tp3, \\_Tp4 >](#)
- struct [\\_\\_gnu\\_pbds::null\\_type](#)
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- struct [\\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)
- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)
- struct [\\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

#### 7.547.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

## 7.548 `tree_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define` [PB\\_DS\\_BRANCH\\_POLICY\\_BASE](#)
- `#define` [PB\\_DS\\_CLASS\\_C\\_DEC](#)
- `#define` [PB\\_DS\\_CLASS\\_T\\_DEC](#)

#### 7.548.1 Detailed Description

Contains tree-related policies.

## 7.549 trie\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- struct [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#)< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

### Variables

- `template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename ↵  
_Alloc>  
detail::integral_constant< int, Reverse > __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max↵  
_E_Val, Reverse, _Alloc >::s_rev_ind`

#### 7.549.1 Detailed Description

Contains trie-related policies.

## 7.550 pod\_char\_traits.h File Reference

### Classes

- struct [std::char\\_traits](#)< [\\_\\_gnu\\_cxx::character](#)< \_Value, \_Int, \_St > >
- struct [\\_\\_gnu\\_cxx::character](#)< \_Value, \_Int, \_St >

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename _Value, typename _Int, typename _St>  
bool \_\_gnu\_cxx::operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St >  
&rhs)`
- `template<typename _Value, typename _Int, typename _St>  
bool \_\_gnu\_cxx::operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St >  
&rhs)`

#### 7.550.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.551 pointer.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Invalid\\_type](#)
- class [\\_\\_gnu\\_cxx::Pointer\\_adapter<\\_Storage\\_policy>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl<const \\_Tp>](#)
- class [\\_\\_gnu\\_cxx::Std\\_pointer\\_impl<\\_Tp>](#)
- struct [\\_\\_gnu\\_cxx::Unqualified\\_type<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Macros

- `#define CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

### Functions

- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const Pointer\_adapter<_Tp2> &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const Pointer\_adapter<_Tp> &__lhs, const Pointer\_adapter<_Tp> &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const Pointer\_adapter<_Tp> &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const Pointer\_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator!= (const Pointer\_adapter<_Tp1> &__lhs, const Pointer\_adapter<_Tp2> &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (int __lhs, const Pointer\_adapter<_Tp> &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const Pointer\_adapter<_Tp2> &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (const Pointer\_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator< (const Pointer\_adapter<_Tp1> &__lhs, const Pointer\_adapter<_Tp2> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT>`  
`std::basic_ostream<_CharT, _Traits> &__gnu_cxx::operator<< (std::basic_ostream<_CharT, _Traits> &__os, const Pointer\_adapter<_StoreT> &__p)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const Pointer\_adapter<_Tp2> &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator<= (const Pointer\_adapter<_Tp> &__lhs, const Pointer\_adapter<_Tp> &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (const Pointer\_adapter<_Tp1> &__lhs, _Tp2 __rhs)`

- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2>`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`

### 7.551.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

## 7.552 pool\_allocator.h File Reference

### Classes

- class `__gnu_cxx::__pool_alloc< _Tp >`
- class `__gnu_cxx::__pool_alloc_base`

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp>`  
`bool \_\_gnu\_cxx::operator== (const \_\_pool\_alloc< _Tp > &, const \_\_pool\_alloc< _Tp > &)`

## Variables

- `template<typename _Tp>`  
`_Atomic_word \_\_gnu\_cxx::\_\_pool\_alloc< _Tp >::\_S\_force\_new`

### 7.552.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.553 [rb\\_tree](#) File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::rb\\_tree](#)< [\\_Key](#), [\\_Value](#), [\\_KeyOfValue](#), [\\_Compare](#), [\\_Alloc](#) >

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define \_RB\_TREE`

### 7.553.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.554 [rc\\_string\\_base.h](#) File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base](#)< [\\_CharT](#), [\\_Traits](#), [\\_Alloc](#) >

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Variables

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`\_\_rc\_string\_base< \_CharT, \_Traits, \_Alloc >::\_Rep\_empty \_\_gnu\_cxx::\_\_rc\_string\_base< \_CharT, \_Traits, \_Alloc >::\_S\_empty\_rep`

### 7.554.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 7.555 rope File Reference

### Classes

- class `__gnu_cxx::rope<_CharT, _Alloc>`

### Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::__detail`
- namespace `std`
- namespace `std::tr1`

### Macros

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`
- `#define __ROPE`

### Typedefs

- typedef `rope<char> __gnu_cxx::crope`
- typedef `rope<wchar_t> __gnu_cxx::wrope`

### Enumerations

- enum { `_S_max_rope_depth` }
- enum `_Tag` { `_S_leaf` , `_S_concat` , `_S_substringfn` , `_S_function` }

### Functions

- `crope::reference __gnu_cxx::__mutable_reference_at (crope &__c, std::size_t __i)`
- `template<typename _ForwardIterator, typename _Allocator>`  
`void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp>`  
`void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator __last, std::allocator<_Tp>)`
- `template<class _CharT>`  
`void __gnu_cxx::__S_cond_store_eos (_CharT &)`
- `void __gnu_cxx::__S_cond_store_eos (char &__c)`
- `void __gnu_cxx::__S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT>`  
`_CharT __gnu_cxx::__S_eos (_CharT *)`
- `template<class _CharT>`  
`bool __gnu_cxx::__S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::__S_is_basic_char_type (char *)`
- `bool __gnu_cxx::__S_is_basic_char_type (wchar_t *)`
- `template<class _CharT>`  
`bool __gnu_cxx::__S_is_one_byte_char_type (_CharT *)`
- `bool __gnu_cxx::__S_is_one_byte_char_type (char *)`

- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`std::ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`



- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc>`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc>`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

## Variables

- `template<class _CharT, class _Alloc>`  
`rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

- `template<class _CharT, class _Alloc>`  
`const rope<_CharT, _Alloc >::size_type __gnu_cxx::rope<_CharT, _Alloc >::npos`

### 7.555.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.556 ropeimpl.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(cond)`
- `#define __N(msgid)`
- `#define __NO_CTYPE`
- `#define _GLIBCXX11_DEPRECATED`
- `#define _GLIBCXX11_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX11_USE_C99_MATH`
- `#define _GLIBCXX11_USE_C99_STDIO`
- `#define _GLIBCXX11_USE_C99_STDLIB`
- `#define _GLIBCXX11_USE_C99_WCHAR`
- `#define _GLIBCXX14_DEPRECATED`
- `#define _GLIBCXX14_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX17_DEPRECATED`
- `#define _GLIBCXX17_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX20_DEPRECATED`
- `#define _GLIBCXX20_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX23_DEPRECATED`
- `#define _GLIBCXX23_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX26_DEPRECATED`
- `#define _GLIBCXX26_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX98_USE_C99_COMPLEX`
- `#define _GLIBCXX98_USE_C99_MATH`
- `#define _GLIBCXX98_USE_C99_STDIO`
- `#define _GLIBCXX98_USE_C99_STDLIB`
- `#define _GLIBCXX98_USE_C99_WCHAR`
- `#define _GLIBCXX_ASSERT_FAIL(_Condition)`
- `#define _GLIBCXX_ATOMIC_WORD_BUILTINS`
- `#define _GLIBCXX_AUTO_CAST(X)`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE`

- `#define _GLIBCXX_CPU_DEFINES`
- `#define _GLIBCXX_DEFAULT_ABI_TAG`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX_DOXYGEN_ONLY(X)`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_END_NAMESPACE_ALGO`
- `#define _GLIBCXX_END_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_END_NAMESPACE_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`
- `#define _GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_VERSION`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`
- `#define _GLIBCXX_FULLY_DYNAMIC_STRING`
- `#define _GLIBCXX_HAS_BUILTIN(B)`
- `#define _GLIBCXX_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_ALIGNED_ALLOC`
- `#define _GLIBCXX_HAVE_ARC4RANDOM`
- `#define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- `#define _GLIBCXX_HAVE_ASINF`
- `#define _GLIBCXX_HAVE_ASINL`
- `#define _GLIBCXX_HAVE_AT_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_ATAN2F`
- `#define _GLIBCXX_HAVE_ATAN2L`
- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATOMIC_LOCK_POLICY`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP`
- `#define _GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE`
- `#define _GLIBCXX_HAVE_BUILTIN_LAUNDER`
- `#define _GLIBCXX_HAVE_C99_FLT_EVAL_TYPES`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DECL_STRNLEN`
- `#define _GLIBCXX_HAVE_DIRFD`
- `#define _GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FDOPENDIR`

- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_FWRITE_UNLOCKED`
- `#define _GLIBCXX_HAVE_GETENTROPY`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_GETS`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONv`
- `#define _GLIBCXX_HAVE_IS_CONSTANT_EVALUATED`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINK`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_LSEEK`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMALIGN`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_O_NONBLOCK`
- `#define _GLIBCXX_HAVE_OPENAT`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POSIX_MEMALIGN`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`

- `#define _GLIBCXX_HAVE_READLINK`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SECURE_GETENV`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SOCKETATMARK`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STACKTRACE`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMLINK`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TIMESPEC_GET`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_TRUNCATE`
- `#define _GLIBCXX_HAVE_UNLINKAT`
- `#define _GLIBCXX_HAVE_USELOCALE`
- `#define _GLIBCXX_HAVE_VFWSCANF`
- `#define _GLIBCXX_HAVE_VSWSCANF`
- `#define _GLIBCXX_HAVE_VWSCANF`
- `#define _GLIBCXX_HAVE_WCSTOF`
- `#define _GLIBCXX_HAVE_WRITEV`
- `#define _GLIBCXX_HOSTED`
- `#define _GLIBCXX_ICONV_CONST`
- `#define _GLIBCXX_INLINE_VERSION`
- `#define _GLIBCXX_LT_OBJDIR`
- `#define _GLIBCXX_MANGLE_SIZE_T`
- `#define _GLIBCXX_MAY_HAVE___CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_NAMESPACE_CXX11`
- `#define _GLIBCXX_NAMESPACE_LDBL`
- `#define _GLIBCXX_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC`
- `#define _GLIBCXX_NODISCARD`
- `#define _GLIBCXX_NOEXCEPT_PARM`
- `#define _GLIBCXX_NOEXCEPT_QUAL`
- `#define _GLIBCXX_OS_DEFINES`

- `#define _GLIBCXX_RELEASE`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STATIC_TZDATA`
- `#define _GLIBCXX_STD_A`
- `#define _GLIBCXX_STD_C`
- `#define _GLIBCXX_STDIO_EOF`
- `#define _GLIBCXX_STDIO_SEEK_CUR`
- `#define _GLIBCXX_STDIO_SEEK_END`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_TXN_SAFE`
- `#define _GLIBCXX_TXN_SAFE_DYN`
- `#define _GLIBCXX_USE_ALLOCATOR_NEW`
- `#define _GLIBCXX_USE_BUILTIN_TRAIT(BT)`
- `#define _GLIBCXX_USE_C11_UCHAR_CXX11`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`
- `#define _GLIBCXX_USE_C99_CTYPE`
- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH_FUNCS`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_CHDIR`
- `#define _GLIBCXX_USE_CHMOD`
- `#define _GLIBCXX_USE_CLOCK_MONOTONIC`
- `#define _GLIBCXX_USE_CLOCK_REALTIME`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_DEV_RANDOM`
- `#define _GLIBCXX_USE_DUAL_ABI`
- `#define _GLIBCXX_USE_FCHMOD`
- `#define _GLIBCXX_USE_FCHMODAT`
- `#define _GLIBCXX_USE_FSEEKO_FTELLO`
- `#define _GLIBCXX_USE_GET_NPROCS`
- `#define _GLIBCXX_USE_GETCWD`
- `#define _GLIBCXX_USE_GETTIMEOFDAY`
- `#define _GLIBCXX_USE_GLIBC_STDIO_EXT`
- `#define _GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE`
- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_LSTAT`
- `#define _GLIBCXX_USE_MKDIR`
- `#define _GLIBCXX_USE_NANOSLEEP`
- `#define _GLIBCXX_USE_NL_LANGINFO_L`
- `#define _GLIBCXX_USE_NLS`

- `#define _GLIBCXX_USE_PROC_SELF_STATUS`
- `#define _GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT`
- `#define _GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK`
- `#define _GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_REALPATH`
- `#define _GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- `#define _GLIBCXX_USE_SCHED_YIELD`
- `#define _GLIBCXX_USE_SENDFILE`
- `#define _GLIBCXX_USE_ST_MTIM`
- `#define _GLIBCXX_USE_STD_SPEC_FUNCS`
- `#define _GLIBCXX_USE_STDIO_LOCKING`
- `#define _GLIBCXX_USE_STRUCT_TM_TM_ZONE`
- `#define _GLIBCXX_USE_TMPNAM`
- `#define _GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20`
- `#define _GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T`
- `#define _GLIBCXX_USE_UTILITY`
- `#define _GLIBCXX_USE_UTILITY`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_VERBOSE`
- `#define _GLIBCXX_VERBOSE_ASSERT`
- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_X86_RDRAND`
- `#define _GLIBCXX_X86_RDSEED`
- `#define _GLIBCXX_ZONEINFO_DIR`
- `#define _GTHREAD_USE_MUTEX_TIMEDLOCK`

## Functions

- `template<class _CharT, class _Traits>`  
`void __gnu_cxx::Rope_fill (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)`
- `template<class _CharT>`  
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator>`  
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc>`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

## Variables

- `template<class _CharT, class _Alloc>`  
`_CharT __gnu_cxx::rope< _CharT, _Alloc >::_S_empty_c_str [1]`
- `template<class _CharT, class _Alloc>`  
`const unsigned long __gnu_cxx::rope< _CharT, _Alloc >::_S_min_len [int(__detail::_S_max_rope_depth)+1]`

### 7.556.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

## 7.557 slist File Reference

### Classes

- class `__gnu_cxx::slist<_Tp, _Alloc>`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Macros

- `#define _SLIST`

### Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link` (`_Slist_node_base * __prev_node`, `_Slist_node_base * __new_node`)
- `_Slist_node_base * __gnu_cxx::__slist_previous` (`_Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `const _Slist_node_base * __gnu_cxx::__slist_previous` (`const _Slist_node_base * __head`, `const _Slist_node_base * __node`)
- `_Slist_node_base * __gnu_cxx::__slist_reverse` (`_Slist_node_base * __node`)
- `std::size_t __gnu_cxx::__slist_size` (`_Slist_node_base * __node`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __before_first`, `_Slist_node_base * __before_last`)
- `void __gnu_cxx::__slist_splice_after` (`_Slist_node_base * __pos`, `_Slist_node_base * __head`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator!=` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator<` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator<=` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator==` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator>` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`bool __gnu_cxx::operator>=` (`const slist<_Tp, _Alloc> &_SL1`, `const slist<_Tp, _Alloc> &_SL2`)
- `template<class _Tp, class _Alloc>`  
`void __gnu_cxx::swap` (`slist<_Tp, _Alloc> &_x`, `slist<_Tp, _Alloc> &_y`)

### 7.557.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).



## 7.558 sso\_string\_base.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.558.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 7.559 stdio\_filebuf.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.559.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.560 stdio\_sync\_filebuf.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

#### 7.560.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.561 string\_conversions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>  
_Ret \_\_gnu\_cxx::\_\_stoa (_TRet>(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const  
_CharT *__str, std::size_t *__idx, _Base... __base)`
- `template<typename _String, typename _CharT = typename _String::value_type>  
_String \_\_gnu\_cxx::\_\_to\_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t  
__n, const _CharT *__fmt,...)`

#### 7.561.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.562 throw\_allocator.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition](#)
- class [\\_\\_gnu\\_cxx::throw\\_allocator\\_base< \\_Tp, \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_limit< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_random< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_base< \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- void [\\_\\_gnu\\_cxx::\\_\\_throw\\_forced\\_error \(\)](#)
- template<typename \_Cond>  
[throw\\_value\\_base< \\_Cond >](#) [\\_\\_gnu\\_cxx::operator\\*](#) (const [throw\\_value\\_base< \\_Cond >](#) &\_\_a, const [throw\\_value\\_base< \\_Cond >](#) &\_\_b)
- template<typename \_Cond>  
[throw\\_value\\_base< \\_Cond >](#) [\\_\\_gnu\\_cxx::operator+](#) (const [throw\\_value\\_base< \\_Cond >](#) &\_\_a, const [throw\\_value\\_base< \\_Cond >](#) &\_\_b)
- template<typename \_Cond>  
[throw\\_value\\_base< \\_Cond >](#) [\\_\\_gnu\\_cxx::operator-](#) (const [throw\\_value\\_base< \\_Cond >](#) &\_\_a, const [throw\\_value\\_base< \\_Cond >](#) &\_\_b)
- template<typename \_Cond>  
bool [\\_\\_gnu\\_cxx::operator<](#) (const [throw\\_value\\_base< \\_Cond >](#) &\_\_a, const [throw\\_value\\_base< \\_Cond >](#) &\_\_b)
- [std::ostream](#) & [\\_\\_gnu\\_cxx::operator<<](#) ([std::ostream](#) &os, const [annotate\\_base](#) &\_\_b)
- template<typename \_Tp, typename \_Cond>  
bool [\\_\\_gnu\\_cxx::operator==](#) (const [throw\\_allocator\\_base< \\_Tp, \\_Cond >](#) &, const [throw\\_allocator\\_base< \\_Tp, \\_Cond >](#) &)
- template<typename \_Cond>  
bool [\\_\\_gnu\\_cxx::operator==](#) (const [throw\\_value\\_base< \\_Cond >](#) &\_\_a, const [throw\\_value\\_base< \\_Cond >](#) &\_\_b)
- template<typename \_Cond>  
void [\\_\\_gnu\\_cxx::swap](#) ([throw\\_value\\_base< \\_Cond >](#) &\_\_a, [throw\\_value\\_base< \\_Cond >](#) &\_\_b)

### 7.562.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

## 7.563 `type_traits.h` File Reference

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Type>`  
`constexpr bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type>`  
`constexpr bool __gnu_cxx::__is_null_pointer (_Type)`
- `constexpr bool __gnu_cxx::__is_null_pointer (std::nullptr_t)`

### 7.563.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.564 `typelist.h` File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `__gnu_cxx::typelist`

### Macros

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TYPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`
- `#define _GLIBCXX_TYPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TYPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`

- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

## Functions

- `template<typename Fn, typename Typelist>`  
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename Typelist>`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist>`  
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>`  
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`

### 7.564.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

## 7.565 vstring.h File Reference

### Classes

- class `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`
- struct `std::hash< __gnu_cxx::__u16vstring >`
- struct `std::hash< __gnu_cxx::__u32vstring >`
- struct `std::hash< __gnu_cxx::__vstring >`
- struct `std::hash< __gnu_cxx::__wvstring >`

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs)`

- Generated by Doxygen

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT`  
`*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >`  
`&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &↵`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT`  
`*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >`  
`&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > &std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string<`  
`_CharT, _Traits, _Alloc, _Base > &__str)`
- `double __gnu_cxx::stod (const _vstring &__str, std::size_t *__idx=0)`
- `double __gnu_cxx::stod (const _wvstring &__str, std::size_t *__idx=0)`
- `float __gnu_cxx::stof (const _vstring &__str, std::size_t *__idx=0)`
- `float __gnu_cxx::stof (const _wvstring &__str, std::size_t *__idx=0)`
- `int __gnu_cxx::stoi (const _vstring &__str, std::size_t *__idx=0, int __base=10)`
- `int __gnu_cxx::stoi (const _wvstring &__str, std::size_t *__idx=0, int __base=10)`
- `long __gnu_cxx::stol (const _vstring &__str, std::size_t *__idx=0, int __base=10)`
- `long __gnu_cxx::stol (const _wvstring &__str, std::size_t *__idx=0, int __base=10)`
- `long double __gnu_cxx::stold (const _vstring &__str, std::size_t *__idx=0)`
- `long double __gnu_cxx::stold (const _wvstring &__str, std::size_t *__idx=0)`
- `long long __gnu_cxx::stoll (const _vstring &__str, std::size_t *__idx=0, int __base=10)`
- `long long __gnu_cxx::stoll (const _wvstring &__str, std::size_t *__idx=0, int __base=10)`
- `unsigned long __gnu_cxx::stoul (const _vstring &__str, std::size_t *__idx=0, int __base=10)`
- `unsigned long __gnu_cxx::stoul (const _wvstring &__str, std::size_t *__idx=0, int __base=10)`
- `unsigned long long __gnu_cxx::stoull (const _vstring &__str, std::size_t *__idx, int __base=10)`
- `unsigned long long __gnu_cxx::stoull (const _wvstring &__str, std::size_t *__idx=0, int __base=10)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `__vstring __gnu_cxx::to_string (double __val)`
- `__vstring __gnu_cxx::to_string (float __val)`
- `__vstring __gnu_cxx::to_string (int __val)`
- `__vstring __gnu_cxx::to_string (long __val)`
- `__vstring __gnu_cxx::to_string (long double __val)`
- `__vstring __gnu_cxx::to_string (long long __val)`
- `__vstring __gnu_cxx::to_string (unsigned __val)`
- `__vstring __gnu_cxx::to_string (unsigned long __val)`
- `__vstring __gnu_cxx::to_string (unsigned long long __val)`
- `__wvstring __gnu_cxx::to_wstring (double __val)`
- `__wvstring __gnu_cxx::to_wstring (float __val)`
- `__wvstring __gnu_cxx::to_wstring (int __val)`
- `__wvstring __gnu_cxx::to_wstring (long __val)`
- `__wvstring __gnu_cxx::to_wstring (long double __val)`
- `__wvstring __gnu_cxx::to_wstring (long long __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned long __val)`
- `__wvstring __gnu_cxx::to_wstring (unsigned long long __val)`

### 7.565.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 7.566 vstring.tcc File Reference

### Namespaces

- namespace `__gnu_cxx`
- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ ( _CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base > & __str)`

## Variables

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _↵  
Traits, _Alloc, _Base >::npos`

### 7.566.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 7.567 vstring\_fwd.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### Typedefs

- `typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > __gnu_↵  
cxx::__rc_string`
- `typedef __vstring __gnu_cxx::__sso_string`
- `typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base  
> __gnu_cxx::__u16rc_string`
- `typedef __u16vstring __gnu_cxx::__u16sso_string`
- `typedef __versa_string< char16_t > __gnu_cxx::__u16vstring`
- `typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base  
> __gnu_cxx::__u32rc_string`
- `typedef __u32vstring __gnu_cxx::__u32sso_string`
- `typedef __versa_string< char32_t > __gnu_cxx::__u32vstring`
- `typedef __versa_string< char > __gnu_cxx::__vstring`
- `typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base >  
__gnu_cxx::__wrc_string`
- `typedef __wvstring __gnu_cxx::__wsso_string`
- `typedef __versa_string< wchar_t > __gnu_cxx::__wvstring`

### 7.567.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 7.568 vstring\_util.h File Reference

### Namespaces

- namespace `__gnu_cxx`

### 7.568.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.



## 7.569 fenv.h File Reference

### 7.569.1 Detailed Description

This is a Standard C++ Library header.

## 7.570 experimental/filesystem File Reference

### Macros

- `#define __cpp_lib_experimental_filesystem`
- `#define _GLIBCXX_EXPERIMENTAL_FILESYSTEM`

### 7.570.1 Detailed Description

This is a TS C++ Library header.

## 7.571 filesystem File Reference

### 7.571.1 Detailed Description

This is a Standard C++ Library header.

## 7.572 flat\_map File Reference

### 7.572.1 Detailed Description

This is a Standard C++ Library header.

## 7.573 flat\_set File Reference

### 7.573.1 Detailed Description

This is a Standard C++ Library header.

## 7.574 format File Reference

### 7.574.1 Detailed Description

This is a Standard C++ Library header.

## 7.575 debug/forward\_list File Reference

### Classes

- class [\\_\\_gnu\\_debug::Safe\\_forward\\_list<\\_SafeSequence >](#)
- class [std::\\_\\_debug::forward\\_list<\\_Tp, \\_Alloc >](#)

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Macros

- `#define __glibcxx_check_valid_fl_range(_First, _Last, _Dist)`
- `#define __GLIBCXX20_ONLY(__expr)`
- `#define __GLIBCXX_DEBUG_FORWARD_LIST`
- `#define __GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG`

## Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
std::__debug::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>  
std::__debug::forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >`
- `template<typename _Tp, typename _Alloc>  
constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>  
bool std::__debug::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc>  
void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

### 7.575.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.576 experimental/forward\_list File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define __GLIBCXX_EXPERIMENTAL_FORWARD_LIST`

### Typedefs

- `template<typename _Tp>  
using std::experimental::fundamentals_v2::pmr::forward_list`

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up>  
void std::experimental::erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>  
void std::experimental::erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 7.576.1 Detailed Description

This is a TS C++ Library header.

## 7.577 forward\_list File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Tp>`  
using **std::pmr::forward\_list**

### 7.577.1 Detailed Description

This is a Standard C++ Library header.

## 7.578 fstream File Reference

### Classes

- class [std::basic\\_filebuf< \\_CharT, \\_Traits >](#)
- class [std::basic\\_fstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ifstream< \\_CharT, \\_Traits >](#)
- class [std::basic\\_ofstream< \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_BUFSIZ`

### Typedefs

- `template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>`  
using **std::\_If\_fs\_path**

### Functions

- `template<class _CharT, class _Traits>`  
void [std::swap](#) ([basic\\_filebuf< \\_CharT, \\_Traits >](#) &\_\_x, [basic\\_filebuf< \\_CharT, \\_Traits >](#) &\_\_y)
- `template<class _CharT, class _Traits>`  
void [std::swap](#) ([basic\\_fstream< \\_CharT, \\_Traits >](#) &\_\_x, [basic\\_fstream< \\_CharT, \\_Traits >](#) &\_\_y)
- `template<class _CharT, class _Traits>`  
void [std::swap](#) ([basic\\_ifstream< \\_CharT, \\_Traits >](#) &\_\_x, [basic\\_ifstream< \\_CharT, \\_Traits >](#) &\_\_y)
- `template<class _CharT, class _Traits>`  
void [std::swap](#) ([basic\\_ofstream< \\_CharT, \\_Traits >](#) &\_\_x, [basic\\_ofstream< \\_CharT, \\_Traits >](#) &\_\_y)

### 7.578.1 Detailed Description

This is a Standard C++ Library header.

## 7.579 experimental/functional File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_boyer_moore_searching`
- `#define __cpp_lib_experimental_not_fn`
- `#define _GLIBCXX_EXPERIMENTAL_FUNCTIONAL`

### Typedefs

- `template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>`  
using [std::experimental::\\_\\_boyer\\_moore\\_base\\_t](#)

### Functions

- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>`  
`boyer_moore_horspool_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::make\_boyer\_moore\_horspool\_searcher`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>`  
`boyer_moore_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::make\_boyer\_moore\_searcher`  
`(_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>`  
`default_searcher<_ForwardIterator, _BinaryPredicate> std::experimental::make\_default\_searcher`  
`(_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Fn>`  
`auto std::experimental::not\_fn (_Fn && __fn) noexcept(std::is\_nothrow\_constructible< std::decay\_t< _Fn >, _Fn && >::value)`

### Variables

- `template<typename _Tp>`  
`constexpr bool std::experimental::is\_bind\_expression\_v`
- `template<typename _Tp>`  
`constexpr int std::experimental::is\_placeholder\_v`

#### 7.579.1 Detailed Description

This is a TS C++ Library header.

## 7.580 ext/functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose](#)< \_Operation1, \_Operation2, \_Operation3 >
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun](#)< \_Result, \_Arg1, \_Arg2 >
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun](#)< \_Result, \_Argument >
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun](#)< \_Result >

- struct [\\_\\_gnu\\_cxx::project1st<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::project2nd<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::select1st<\\_Pair >](#)
- struct [\\_\\_gnu\\_cxx::select2nd<\\_Pair >](#)
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose<\\_Operation1, \\_Operation2 >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Macros

- `#define` [\\_EXT\\_FUNCTIONAL](#)

## Functions

- [template<class \\_Operation1, class \\_Operation2>](#)  
[unary\\_compose<\\_Operation1, \\_Operation2 > \\_\\_gnu\\_cxx::compose1](#) (const \_Operation1 &\_\_fn1, const \_↵  
Operation2 &\_\_fn2)
- [template<class \\_Operation1, class \\_Operation2, class \\_Operation3>](#)  
[binary\\_compose<\\_Operation1, \\_Operation2, \\_Operation3 > \\_\\_gnu\\_cxx::compose2](#) (const \_Operation1 &\_\_fn1,  
const \_Operation2 &\_\_fn2, const \_Operation3 &\_\_fn3)
- [template<class \\_Result>](#)  
[constant\\_void\\_fun<\\_Result > \\_\\_gnu\\_cxx::constant0](#) (const \_Result &\_\_val)
- [template<class \\_Result>](#)  
[constant\\_unary\\_fun<\\_Result, \\_Result > \\_\\_gnu\\_cxx::constant1](#) (const \_Result &\_\_val)
- [template<class \\_Result>](#)  
[constant\\_binary\\_fun<\\_Result, \\_Result, \\_Result > \\_\\_gnu\\_cxx::constant2](#) (const \_Result &\_\_val)
- [template<class \\_Tp>](#)  
[\\_Tp \\_\\_gnu\\_cxx::identity\\_element](#) (std::multiplies<\_Tp >)
- [template<class \\_Tp>](#)  
[\\_Tp \\_\\_gnu\\_cxx::identity\\_element](#) (std::plus<\_Tp >)
- [template<class \\_Ret, class \\_Tp, class \\_Arg>](#)  
[std::const\\_mem\\_fun1\\_t<\\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg) const)
- [template<class \\_Ret, class \\_Tp, class \\_Arg>](#)  
[std::mem\\_fun1\\_t<\\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))
- [template<class \\_Ret, class \\_Tp, class \\_Arg>](#)  
[std::const\\_mem\\_fun1\\_ref\\_t<\\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1\\_ref](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg) const)
- [template<class \\_Ret, class \\_Tp, class \\_Arg>](#)  
[std::mem\\_fun1\\_ref\\_t<\\_Ret, \\_Tp, \\_Arg > \\_\\_gnu\\_cxx::mem\\_fun1\\_ref](#) (\_Ret(\_Tp::\*\_\_f)(\_Arg))

### 7.580.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 7.581 functional File Reference

### Classes

- class [std::\\_Not\\_fn<\\_Fn >](#)
- struct [std::\\_Placeholder<\\_Num >](#)
- struct [std::is\\_bind\\_expression<\\_Tp >](#)
- struct [std::is\\_bind\\_expression<\\_Bind<\\_Signature > >](#)

- struct `std::is_bind_expression< _Bind_result< _Result, _Signature > >`
- struct `std::is_bind_expression< const _Bind< _Signature > >`
- struct `std::is_bind_expression< const _Bind_result< _Result, _Signature > >`
- struct `std::is_bind_expression< const volatile _Bind< _Signature > >`
- struct `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`
- struct `std::is_bind_expression< volatile _Bind< _Signature > >`
- struct `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`
- struct `std::is_placeholder< _Tp >`
- struct `std::is_placeholder< _Placeholder< _Num > >`

## Namespaces

- namespace `std`
- namespace `std::placeholders`

## Macros

- `#define _GLIBCXX_NOT_FN_CALL_OP(_QUALS)`
- `#define _GLIBCXX_PLACEHOLDER`

## Typedefs

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
using `std::__is_socketlike`

## Functions

- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind ( _Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Func, typename... _BoundArgs>`  
`constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind ( _Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Tp, typename _Class>`  
`constexpr _Mem_fn< _Tp _Class::* > std::mem_fn ( _Tp _Class::*__pm) noexcept`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`
- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`

- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`
- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`
- `template<typename _Tp>`  
`constexpr bool std::is_bind_expression_v`
- `template<typename _Tp>`  
`constexpr int std::is_placeholder_v`

### 7.581.1 Detailed Description

This is a Standard C++ Library header.

## 7.582 future File Reference

### Classes

- class `std::__basic_future< _Res >`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

### Namespaces

- namespace `std`

### Enumerations

- enum class `std::future_errc` { `future_already_retrieved` , `promise_already_satisfied` , `no_state` , `broken_←  
promise` }
- enum class `std::future_status` { `ready` , `timeout` , `deferred` }
- enum class `std::launch` { `async` , `deferred` }

## Functions

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `const error_category & std::future_category () noexcept`
- `error_code std::make_error_code (future_errc __errc) noexcept`
- `error_condition std::make_error_condition (future_errc __errc) noexcept`
- `constexpr launch std::operator& (launch __x, launch __y) noexcept`
- `constexpr launch & std::operator&= (launch &__x, launch __y) noexcept`
- `constexpr launch std::operator^ (launch __x, launch __y) noexcept`
- `constexpr launch & std::operator^= (launch &__x, launch __y) noexcept`
- `constexpr launch std::operator| (launch __x, launch __y) noexcept`
- `constexpr launch & std::operator|= (launch &__x, launch __y) noexcept`
- `constexpr launch std::operator~ (launch __x) noexcept`
- `template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>`  
`std::packaged_task (_Fun) -> packaged_task< _Signature >`
- `template<typename _Res, typename... _ArgTypes>`  
`std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>`
- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept`
- `template<typename _Res>`  
`void std::swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`

### 7.582.1 Detailed Description

This is a Standard C++ Library header.

## 7.583 generator File Reference

### 7.583.1 Detailed Description

This is a Standard C++ Library header.

## 7.584 inplace\_vector File Reference

### Namespaces

- namespace `std`

## Functions

- `template<typename _Tp, size_t _Nm, typename _Up = _Tp>`  
`constexpr size_t std::erase (::inplace_vector< _Tp, _Nm > &__cont, const _Up &__value)`
- `template<typename _Tp, size_t _Nm, typename _Predicate>`  
`constexpr size_t std::erase_if (::inplace_vector< _Tp, _Nm > &__cont, _Predicate __pred)`

### 7.584.1 Detailed Description

This is a Standard C++ Library header.



## 7.585 iomanip File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _MoneyT>  
_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT>  
_Get_time< _CharT > std::get\_time (std::tm *__tmb, const _CharT *__fmt)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_↵  
money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Put_time<↵  
_CharT > __f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _↵  
Resetiosflags __f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setbase↵  
__f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setfill<↵  
_CharT > __f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setiosflags↵  
__f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _↵  
Setprecision __f)`
- `template<typename _CharT, typename _Traits>  
basic\_ostream< _CharT, _Traits > & std::operator<< (basic\_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template ostream & std::operator<< (ostream &, _Resetiosflags)`
- `template ostream & std::operator<< (ostream &, _Setbase)`
- `template ostream & std::operator<< (ostream &, _Setfill< char >)`
- `template ostream & std::operator<< (ostream &, _Setiosflags)`
- `template ostream & std::operator<< (ostream &, _Setprecision)`
- `template ostream & std::operator<< (ostream &, _Setw)`
- `template wostream & std::operator<< (wostream &, _Resetiosflags)`
- `template wostream & std::operator<< (wostream &, _Setbase)`
- `template wostream & std::operator<< (wostream &, _Setfill< wchar_t >)`
- `template wostream & std::operator<< (wostream &, _Setiosflags)`
- `template wostream & std::operator<< (wostream &, _Setprecision)`
- `template wostream & std::operator<< (wostream &, _Setw)`
- `template<typename _CharT, typename _Traits, typename _MoneyT>  
basic\_istream< _CharT, _Traits > & std::operator>> (basic\_istream< _CharT, _Traits > &__is, _Get_money<↵  
_MoneyT > __f)`
- `template<typename _CharT, typename _Traits>  
basic\_istream< _CharT, _Traits > & std::operator>> (basic\_istream< _CharT, _Traits > &__is, _Get_time<↵  
_CharT > __f)`

- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _↵`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template istream & std::operator>> (istream &, _Resetiosflags)`
- `template istream & std::operator>> (istream &, _Setbase)`
- `template istream & std::operator>> (istream &, _Setfill< char >)`
- `template istream & std::operator>> (istream &, _Setiosflags)`
- `template istream & std::operator>> (istream &, _Setprecision)`
- `template istream & std::operator>> (istream &, _Setw)`
- `template wistream & std::operator>> (wistream &, _Resetiosflags)`
- `template wistream & std::operator>> (wistream &, _Setbase)`
- `template wistream & std::operator>> (wistream &, _Setfill< wchar_t >)`
- `template wistream & std::operator>> (wistream &, _Setiosflags)`
- `template wistream & std::operator>> (wistream &, _Setprecision)`
- `template wistream & std::operator>> (wistream &, _Setw)`
- `template<typename _MoneyT>`  
`_Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT>`  
`_Put_time< _CharT > std::put_time (const std::tm * __tmb, const _CharT * __fmt)`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT>`  
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

### 7.585.1 Detailed Description

This is a Standard C++ Library header.

## 7.586 ios File Reference

### 7.586.1 Detailed Description

This is a Standard C++ Library header.

## 7.587 iosfwd File Reference

### 7.587.1 Detailed Description

This is a Standard C++ Library header.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 7.588 iostream File Reference

### Namespaces

- namespace [std](#)

### Variables

- static ios\_base::Init **std::\_\_ioinit**

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <https://←gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- [ostream std::cerr](#)
- [istream std::cin](#)
- [ostream std::clog](#)
- [ostream std::cout](#)
- [wostream std::wcerr](#)
- [wistream std::wcin](#)
- [wostream std::wclog](#)
- [wostream std::wcout](#)

### 7.588.1 Detailed Description

This is a Standard C++ Library header.

## 7.589 istream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istream<\\_CharT, \\_Traits>::sentry](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Is, typename _Tp>`  
using **std::\_\_rvalue\_stream\_extraction\_t**

## Functions

- `template<typename _CharT, typename _Traits>`  
`void std::__istream_extract (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)`
- `void std::__istream_extract (istream &, char *, streamsize)`
- `template<typename _Istream, typename _Tp>`  
`__rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> ( _Istream &&__is, _Tp &&__x)`
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
  
- `template<typename _CharT, typename _Traits>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits>`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`
- `template<class _Traits>`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
  
- `template<typename _CharT, typename _Traits, size_t _Num>`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_Num])`
- `template<class _Traits, size_t _Num>`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_Num])`

### 7.589.1 Detailed Description

This is a Standard C++ Library header.

## 7.590 experimental/iterator File Reference

### Classes

- class `std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >`

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define __cpp_lib_experimental_ostream_joiner`
- `#define _GLIBCXX_EXPERIMENTAL_ITERATOR`

## Functions

- `template<typename _CharT, typename _Traits, typename _DelimT>`  
`ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::make_ostream_joiner (basic_ostream< _CharT, _Traits > &__os, _DelimT &&__delimiter)`

### 7.590.1 Detailed Description

This is a TS C++ Library header.

## 7.591 ext/iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_ITERATOR`

### Functions

- `template<typename _InputIterator, typename _Distance>`  
`void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance>`  
`void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance>`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

### 7.591.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.592 iterator File Reference

### 7.592.1 Detailed Description

This is a Standard C++ Library header.

## 7.593 latch File Reference

### 7.593.1 Detailed Description

This is a Standard C++ Library header.

## 7.594 limits File Reference

### Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)
- struct [std::numeric\\_limits< \\_Tp >](#)
- struct [std::numeric\\_limits< bool >](#)
- struct [std::numeric\\_limits< char >](#)
- struct [std::numeric\\_limits< char16\\_t >](#)
- struct [std::numeric\\_limits< char32\\_t >](#)
- struct [std::numeric\\_limits< double >](#)
- struct [std::numeric\\_limits< float >](#)
- struct [std::numeric\\_limits< int >](#)
- struct [std::numeric\\_limits< long >](#)
- struct [std::numeric\\_limits< long double >](#)
- struct [std::numeric\\_limits< long long >](#)

- struct [std::numeric\\_limits< short >](#)
- struct [std::numeric\\_limits< signed char >](#)
- struct [std::numeric\\_limits< unsigned char >](#)
- struct [std::numeric\\_limits< unsigned int >](#)
- struct [std::numeric\\_limits< unsigned long >](#)
- struct [std::numeric\\_limits< unsigned long long >](#)
- struct [std::numeric\\_limits< unsigned short >](#)
- struct [std::numeric\\_limits< wchar\\_t >](#)

## Namespaces

- namespace [std](#)

## Macros

- `#define __glibcxx_concat3(P, M, S)`
- `#define __glibcxx_concat3_(P, M, S)`
- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_digits10_b(T, B)`
- `#define __glibcxx_digits_b(T, B)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_n(BITSIZE)`
- `#define __glibcxx_float_tinyness_before`
- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_b(T, B)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_min_b(T, B)`
- `#define __glibcxx_signed(T)`
- `#define __glibcxx_signed_b(T, B)`
- `#define __INT_N(TYPE, BITSIZE, EXT, UEXT)`
- `#define __INT_N_201103(TYPE)`
- `#define __INT_N_U201103(TYPE)`
- `#define __max_digits10`

## Enumerations

- enum [std::float\\_denorm\\_style](#) { [std::denorm\\_indeterminate](#) , [std::denorm\\_absent](#) , [std::denorm\\_present](#) }
- enum [std::float\\_round\\_style](#) { [std::round\\_indeterminate](#) , [std::round\\_toward\\_zero](#) , [std::round\\_to\\_nearest](#) , [std::round\\_toward\\_infinity](#) , [std::round\\_toward\\_neg\\_infinity](#) }

### 7.594.1 Detailed Description

This is a Standard C++ Library header.

## 7.595 debug/list File Reference

### Classes

- class `std::__debug::list<_Tp, _Allocator>`

### Namespaces

- namespace `__gnu_debug`
- namespace `std`
- namespace `std::__debug`

### Macros

- `#define _GLIBCXX20_ONLY(__expr)`
- `#define _GLIBCXX_DEBUG_LIST`
- `#define _GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG`

### Functions

- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list<_ValT, _Allocator>`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::list (size_t, _Tp, _Allocator=_Allocator()) -> list<_Tp, _Allocator>`
- `template<typename _Tp, typename _Alloc>`  
`constexpr __detail::__synth3way_t<_Tp> std::__debug::operator<=> (const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::__debug::operator== (const list<_Tp, _Alloc> &__lhs, const list<_Tp, _Alloc> &__rhs)`
- `template<typename _Tp, typename _Alloc>`  
`void std::__debug::swap (list<_Tp, _Alloc> &__lhs, list<_Tp, _Alloc> &__rhs) noexcept(/*conditional */)`

### 7.595.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.596 experimental/list File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_LIST`

### Typedefs

- `template<typename _Tp>`  
`using std::experimental::fundamentals_v2::pmr::list`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up>`  
`void std::experimental::erase (list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (list< _Tp, _Alloc > &__cont, _Predicate __pred)`

### 7.596.1 Detailed Description

This is a TS C++ Library header.

## 7.597 list File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Tp>`  
`using std::pmr::list`

### 7.597.1 Detailed Description

This is a Standard C++ Library header.

## 7.598 locale File Reference

### Macros

- `#define _GLIBCXX_LOCALE`

### 7.598.1 Detailed Description

This is a Standard C++ Library header.

## 7.599 debug/map File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_MAP`

### 7.599.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.600 experimental/map File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)



## Macros

- `#define _GLIBCXX_EXPERIMENTAL_MAP`

## Typedefs

- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::map`
- `template<typename _Key, typename _Tp, typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::multimap`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
void std::experimental::erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>  
void std::experimental::erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)`

### 7.600.1 Detailed Description

This is a TS C++ Library header.

## 7.601 map File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>  
using std::pmr::map`
- `template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>  
using std::pmr::multimap`

### 7.601.1 Detailed Description

This is a Standard C++ Library header.

## 7.602 math.h File Reference

### Functions

- `template<typename _Tp>  
_Tp abs (const complex< _Tp > &)`
- `template<typename _Tp>  
std::complex< _Tp > acos (const std::complex< _Tp > &)`
- `template<typename _Tp>  
std::complex< _Tp > acosh (const std::complex< _Tp > &)`
- `template<typename _Tp>  
std::complex< _Tp > asin (const std::complex< _Tp > &)`
- `template<typename _Tp>  
std::complex< _Tp > asinh (const std::complex< _Tp > &)`
- `template<typename _Tp>  
std::complex< _Tp > atan (const std::complex< _Tp > &)`

- constexpr float **atan2** (float \_\_y, float \_\_x)
- template<typename \_Tp>  
std::complex< \_Tp > **atanh** (const std::complex< \_Tp > &)
- constexpr float **cbrt** (float \_\_x)
- constexpr float **ceil** (float \_\_x)
- constexpr float **copysign** (float \_\_x, float \_\_y)
- template<typename \_Tp>  
complex< \_Tp > **cos** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **cosh** (const complex< \_Tp > &)
- constexpr float **erf** (float \_\_x)
- constexpr float **erfc** (float \_\_x)
- template<typename \_Tp>  
complex< \_Tp > **exp** (const complex< \_Tp > &)
- constexpr float **exp2** (float \_\_x)
- constexpr float **expm1** (float \_\_x)
- template<typename \_Tp>  
\_Tp **fabs** (const std::complex< \_Tp > &\_\_z)
- constexpr float **fdim** (float \_\_x, float \_\_y)
- constexpr float **floor** (float \_\_x)
- constexpr float **fma** (float \_\_x, float \_\_y, float \_\_z)
- constexpr float **fmax** (float \_\_x, float \_\_y)
- constexpr float **fmin** (float \_\_x, float \_\_y)
- constexpr float **fmod** (float \_\_x, float \_\_y)
- constexpr int **fpclassify** (float \_\_x)
- float **frexp** (float \_\_x, int \*\_\_exp)
- constexpr float **hypot** (float \_\_x, float \_\_y)
- constexpr int **ilogb** (float \_\_x)
- constexpr bool **isfinite** (float \_\_x)
- constexpr bool **isgreater** (float \_\_x, float \_\_y)
- constexpr bool **isgreaterequal** (float \_\_x, float \_\_y)
- constexpr bool **isinf** (float \_\_x)
- constexpr bool **isless** (float \_\_x, float \_\_y)
- constexpr bool **islessequal** (float \_\_x, float \_\_y)
- constexpr bool **islessgreater** (float \_\_x, float \_\_y)
- constexpr bool **isnan** (float \_\_x)
- constexpr bool **isnormal** (float \_\_x)
- constexpr bool **isunordered** (float \_\_x, float \_\_y)
- constexpr float **ldexp** (float \_\_x, int \_\_exp)
- constexpr float **lgamma** (float \_\_x)
- constexpr long long **llrint** (float \_\_x)
- constexpr long long **llround** (float \_\_x)
- template<typename \_Tp>  
complex< \_Tp > **log** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **log10** (const complex< \_Tp > &)
- constexpr float **log1p** (float \_\_x)
- constexpr float **log2** (float \_\_x)
- constexpr float **logb** (float \_\_x)
- constexpr long **lrint** (float \_\_x)
- constexpr long **lround** (float \_\_x)

- float **modf** (float \_\_x, float \*\_\_iptr)
- constexpr float **nearbyint** (float \_\_x)
- constexpr float **nextafter** (float \_\_x, float \_\_y)
- constexpr float **nexttoward** (float \_\_x, long double \_\_y)
- template<typename \_Tp>  
complex< \_Tp > **pow** (const complex< \_Tp > &, int)
- constexpr float **remainder** (float \_\_x, float \_\_y)
- float **remquo** (float \_\_x, float \_\_y, int \*\_\_pquo)
- constexpr float **rint** (float \_\_x)
- constexpr float **round** (float \_\_x)
- constexpr float **scalbln** (float \_\_x, long \_\_ex)
- constexpr float **scalbn** (float \_\_x, int \_\_ex)
- constexpr bool **signbit** (float \_\_x)
- template<typename \_Tp>  
complex< \_Tp > **sin** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **sinh** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **sqrt** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **tan** (const complex< \_Tp > &)
- template<typename \_Tp>  
complex< \_Tp > **tanh** (const complex< \_Tp > &)
- constexpr float **tgamma** (float \_\_x)
- constexpr float **trunc** (float \_\_x)

### 7.602.1 Detailed Description

This is a Standard C++ Library header.

### 7.602.2 Function Documentation

#### abs()

```
template<typename _Tp>
_Tp std::abs (
 const complex< _Tp > & __z) [inline]
```

Return magnitude of z.

#### cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
 const complex< _Tp > & __z) [inline]
```

Return complex cosine of z.

#### cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic cosine of z.

**exp()**

```
template<typename _Tp>
complex< _Tp > std::exp (
 const complex< _Tp > & __z) [inline]
```

Return complex base e exponential of z.

**fabs()**

```
template<typename _Tp>
_Tp std::fabs (
 const std::complex< _Tp > & __z) [inline]
```

**fabs(\_\_z)** TR1 8.1.8 [tr.c99.cmplx.fabs]

**log()**

```
template<typename _Tp>
complex< _Tp > std::log (
 const complex< _Tp > & __z) [inline]
```

Return complex natural logarithm of z.

**log10()**

```
template<typename _Tp>
complex< _Tp > std::log10 (
 const complex< _Tp > & __z) [inline]
```

Return complex base 10 logarithm of z.

**pow()**

```
template<typename _Tp>
complex< _Tp > std::pow (
 const complex< _Tp > & __z,
 int __n) [inline]
```

Return x to the *y*<sup>th</sup> power.

**sin()**

```
template<typename _Tp>
complex< _Tp > std::sin (
 const complex< _Tp > & __z) [inline]
```

Return complex sine of z.

**sinh()**

```
template<typename _Tp>
complex< _Tp > std::sinh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic sine of z.

**sqrt()**

```
template<typename _Tp>
complex< _Tp > std::sqrt (
 const complex< _Tp > & __z) [inline]
```

Return complex square root of z.

**tan()**

```
template<typename _Tp>
complex< _Tp > std::tan (
 const complex< _Tp > & __z) [inline]
```

Return complex tangent of z.

**tanh()**

```
template<typename _Tp>
complex< _Tp > std::tanh (
 const complex< _Tp > & __z) [inline]
```

Return complex hyperbolic tangent of z.

**7.603 mdspan File Reference****Namespaces**

- namespace [std](#)

**Typedefs**

- template<typename \_Slice>  
using **std::\_\_mdspan::\_\_full\_extent\_t**
- template<typename \_IndexType, size\_t \_Rank>  
using **std::dextents**
- template<size\_t \_Rank, typename \_IndexType = size\_t>  
using **std::dims**

**Enumerations**

- enum class **\_LayoutSide** { **\_\_left** , **\_\_right** , **\_\_unknown** }
- enum class **\_SliceKind** { **\_\_extent\_slice** , **\_\_unit\_stride\_slice** , **\_\_full** , **\_\_collapsing** }

**Functions**

- constexpr bool **std::\_\_mdspan::\_\_all\_dynamic** (std::span< const size\_t > \_\_extents)
- constexpr bool **std::\_\_mdspan::\_\_all\_static** (std::span< const size\_t > \_\_extents)
- template<typename \_IndexType, size\_t... \_Counts>  
auto **std::\_\_mdspan::\_\_build\_dextents\_type** ([integer\\_sequence](#)< size\_t, \_Counts... >) -> extents< \_IndexType, ((void) \_Counts, dynamic\_extent)... >
- template<typename \_IndexType, typename \_Slice>  
constexpr auto **std::\_\_mdspan::\_\_canonical\_index** (\_Slice && \_\_slice)
- template<typename \_IndexType, typename \_OffsetType, typename \_SpanType, typename \_StrideType>  
constexpr auto **std::\_\_mdspan::\_\_canonical\_range\_slice** (\_OffsetType \_\_offset, \_SpanType \_\_span, \_StrideType \_\_stride)
- template<typename \_IndexType, size\_t \_Extent, typename \_OIndexType>  
constexpr void **std::\_\_mdspan::\_\_check\_inrange\_index** (const extents< \_IndexType, \_Extent > & \_\_ext, const \_OIndexType & \_\_idx)
- template<typename \_IndexType, size\_t \_Extent, typename \_OIndexType>  
constexpr void **std::\_\_mdspan::\_\_check\_valid\_index** (const extents< \_IndexType, \_Extent > & \_\_ext, const \_OIndexType & \_\_idx)
- template<typename \_IndexType, size\_t \_Extent, typename \_Slice>  
constexpr void **std::\_\_mdspan::\_\_check\_valid\_slice** (const extents< \_IndexType, \_Extent > & \_\_ext, const \_Slice & \_\_slice)

- `template<typename _Extents, typename... _Slices>`  
`constexpr void std::__mdspan::__check_valid_slices (const _Extents &__exts, const _Slices &... __slices)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool std::__mdspan::__contains_zero (const array< _Tp, _Nm > &__exts) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool std::__mdspan::__contains_zero (span< _Tp, _Nm > __exts) noexcept`
- `template<typename _Extents>`  
`constexpr span< const typename _Extents::index_type > std::__mdspan::__dynamic_extents (const _Extents &__exts, size_t __begin=0, size_t __end=_Extents::rank()) noexcept`
- `template<size_t _K, typename _Extents, typename _Slice>`  
`constexpr _Extents::index_type std::__mdspan::__dynamic_slice_extent (const _Extents &__exts, _Slice __slice)`
- `template<typename _Extents>`  
`constexpr bool std::__mdspan::__empty (const _Extents &__exts) noexcept`
- `template<typename _Extents>`  
`constexpr _Extents::index_type std::__mdspan::__extents_prod (const _Extents &__exts, size_t __sta_prod, size_t __begin, size_t __end) noexcept`
- `template<size_t _Index, typename _Extents>`  
`constexpr auto std::__mdspan::__extract_extent (const _Extents &__exts)`
- `template<typename _Extents>`  
`constexpr _Extents::index_type std::__mdspan::__fwd_prod (const _Extents &__exts, size_t __begin, size_t __end) noexcept`
- `template<typename _Extents>`  
`constexpr _Extents::index_type std::__mdspan::__fwd_prod (const _Extents &__exts, size_t __r) noexcept`
- `template<typename _IndexType, size_t _Nm>`  
`constexpr _IndexType std::__mdspan::__fwd_prod (span< const _IndexType, _Nm > __values)`
- `template<typename _PaddedMapping>`  
`constexpr size_t std::__mdspan::__get_static_stride ()`
- `template<typename _IndexType, typename _OIndexTypeRef>`  
`constexpr _IndexType std::__mdspan::__index_type_cast (_OIndexTypeRef &&__other)`
- `template<typename _IndexType, typename... _Slices>`  
`constexpr auto std::__mdspan::__inv_map_rank ()`
- `constexpr bool std::__mdspan::__is_block (span< const _SliceKind > __slice_kinds, size_t __block_size)`
- `template<typename _Extents>`  
`constexpr bool std::__mdspan::__is_representable_extents (const _Extents &__exts) noexcept`
- `template<typename _IndexType, typename _OIndexType>`  
`constexpr bool std::__mdspan::__is_representable_integer (_OIndexType __value)`
- `template<typename _IndexType>`  
`constexpr bool std::__mdspan::__is_representable_least_multiple (size_t __x, size_t __y)`
- `template<size_t _PaddedStride, typename _Extents, typename _LayoutTraits>`  
`constexpr bool std::__mdspan::__is_representable_padded_size ()`
- `constexpr size_t std::__mdspan::__least_multiple (size_t __x, size_t __y)`
- `template<typename _Extents, typename... _Indices>`  
`constexpr _Extents::index_type std::__mdspan::__linear_index_left (const _Extents &__exts, _Indices... __indices) noexcept`
- `template<typename _Extents, typename _Stride, typename... _Indices>`  
`constexpr _Extents::index_type std::__mdspan::__linear_index_leftpad (const _Extents &__exts, _Stride __stride, _Indices... __indices)`
- `template<typename _Extents, typename... _Indices>`  
`constexpr _Extents::index_type std::__mdspan::__linear_index_right (const _Extents &__exts, _Indices... __indices) noexcept`

- `template<typename _Extents, typename _Stride, typename... _Indices>`  
`constexpr _Extents::index_type std::mdspan::linear_index_rightpad (const _Extents &__exts, _Stride &__stride, _Indices... __indices)`
- `template<typename _Mapping, typename... _Indices>`  
`constexpr _Mapping::index_type std::mdspan::linear_index_strides (const _Mapping &__m, _Indices... __indices) noexcept`
- `template<typename _Slice>`  
`constexpr _SliceKind std::mdspan::make_slice_kind ()`
- `template<typename... _Slices>`  
`constexpr array< _SliceKind, sizeof...( _Slices)> std::mdspan::make_slice_kind_array ()`
- `template<typename _Mapping>`  
`constexpr LayoutSide std::mdspan::mapping_side ()`
- `template<typename _Mapping>`  
`constexpr _Mapping::index_type std::mdspan::offset (const _Mapping &__m) noexcept`
- `template<_LayoutSide _Side, size_t _Nm>`  
`static constexpr size_t std::mdspan::padded_block_begin (span< const _SliceKind, _Nm > __slice, __kinds, size_t __sub_rank)`
- `static constexpr size_t std::mdspan::padded_block_begin_generic (span< const _SliceKind > __slice, __kinds, size_t __sub_rank)`
- `template<typename _Extents>`  
`constexpr _Extents::index_type std::mdspan::rev_prod (const _Extents &__exts, size_t __r) noexcept`
- `template<typename _Extents>`  
`constexpr _Extents::index_type std::mdspan::size (const _Extents &__exts) noexcept`
- `template<typename _Slice>`  
`constexpr auto std::mdspan::slice_begin (_Slice __slice)`
- `template<typename _IndexType, typename _Slice>`  
`constexpr auto std::mdspan::slice_cast (_Slice && __slice)`
- `template<typename _Extents>`  
`constexpr const array< size_t, _Extents::rank()> & std::mdspan::static_extents () noexcept`
- `template<typename _Extents>`  
`constexpr span< const size_t > std::mdspan::static_extents (size_t __begin, size_t __end) noexcept`
- `template<typename _Extents, typename _IndexType = typename _Extents::index_type>`  
`constexpr _IndexType std::mdspan::static_quotient (_IndexType __nom=__gnu_cxx::int_traits< __IndexType >::__max)`
- `template<typename _IndexType>`  
`constexpr _IndexType std::mdspan::static_quotient (std::span< const size_t > __sta_exts, _IndexType __nom=__gnu_cxx::int_traits< _IndexType >::__max)`
- `template<typename _IndexType, size_t _Extent, typename _Slice>`  
`constexpr size_t std::mdspan::static_slice_extent ()`
- `template<typename _IndexType, size_t... _Extents, typename... _Slices>`  
`constexpr auto std::mdspan::subextents (const extents< _IndexType, _Extents... > &__exts, _Slices... __slices)`
- `template<typename _Mapping, typename... _Slices>`  
`constexpr auto std::mdspan::submapping (const _Mapping &__mapping, _Slices... __slices)`
- `template<typename _Mapping>`  
`constexpr auto std::mdspan::submdspan_mapping_impl (const _Mapping &__mapping)`
- `template<typename _Mapping, typename... _Slices>`  
`constexpr auto std::mdspan::submdspan_mapping_impl (const _Mapping &__mapping, _Slices... __slices)`
- `template<typename _Mapping, typename... _Slices>`  
`constexpr size_t std::mdspan::suboffset (const _Mapping &__mapping, const _Slices &... __slices)`
- `template<typename _IndexType, typename... _Slices>`  
`constexpr auto std::mdspan::subrank ()`

- `template<typename _SubExts, typename _Mapping, typename... _Slices>`  
`constexpr auto std::__mdspan::__substrides (const _Mapping &__mapping, const _Slices &... __slices)`
- `template<typename _SubExts, typename _Mapping, typename... _Slices>`  
`constexpr auto std::__mdspan::__substrides_generic (const _Mapping &__mapping, const _Slices &... __slices)`
- `template<typename _SubExts, typename _Mapping, typename... _Slices>`  
`constexpr auto std::__mdspan::__substrides_standardized (const _Mapping &__mapping, const _Slices &... __slices)`
- `template<typename _IndexType, size_t... _Extents, typename... _RawSlices>`  
`constexpr auto std::canonical_slices (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)`
- `template<typename... _Integrals>`  
`std::extents ( _Integrals...) -> extents< size_t, __detail::__maybe_static_ext< _Integrals >... >`
- `template<typename _CArray>`  
`std::mdspan ( _CArray &) -> mdspan< remove_all_extents_t< _CArray >, extents< size_t, extent_v< _CArray, 0 > > >`
- `template<typename _ElementType, typename... _Integrals>`  
`std::mdspan ( _ElementType *, _Integrals...) -> mdspan< _ElementType, extents< size_t, __detail::__maybe_static_ext< _Integrals >... > >`
- `template<typename _ElementType, typename _MappingType>`  
`std::mdspan ( _ElementType *, const _MappingType &) -> mdspan< _ElementType, typename _MappingType::extents_type, typename _MappingType::layout_type >`
- `template<typename _ElementType, typename _OIndexType, size_t _Nm>`  
`std::mdspan ( _ElementType *, const array< _OIndexType, _Nm > &) -> mdspan< _ElementType, dextents< size_t, _Nm > >`
- `template<typename _ElementType, typename _IndexType, size_t... _ExtentsPack>`  
`std::mdspan ( _ElementType *, const extents< _IndexType, _ExtentsPack... > &) -> mdspan< _ElementType, extents< _IndexType, _ExtentsPack... > >`
- `template<typename _ElementType, typename _OIndexType, size_t _Nm>`  
`std::mdspan ( _ElementType *, span< _OIndexType, _Nm >) -> mdspan< _ElementType, dextents< size_t, _Nm > >`
- `template<typename _Pointer>`  
`std::mdspan ( _Pointer &&) -> mdspan< remove_pointer_t< remove_reference_t< _Pointer > >, extents< size_t > >`
- `template<typename _MappingType, typename _AccessorType>`  
`std::mdspan (typename _AccessorType::data_handle_type, const _MappingType &, const _AccessorType &) -> mdspan< typename _AccessorType::element_type, typename _MappingType::extents_type, typename _MappingType::layout_type, _AccessorType >`
- `template<typename _IndexType, size_t... _Extents, typename... _RawSlices>`  
`constexpr auto std::subextents (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)`
- `template<typename _ElementType, typename _Extents, typename _Layout, typename _Accessor, typename... _RawSlices>`  
`constexpr auto std::submdspan (const mdspan< _ElementType, _Extents, _Layout, _Accessor > &__md, __raw_slices... __raw_slices)`
- `void std::__mdspan::submdspan_mapping ()=delete`

## Variables

- `template<array _Extents>`  
`constexpr auto std::__mdspan::__fwd_partial_prods`
- `template<typename _Tp>`  
`constexpr bool std::__mdspan::__is_extent_slice`



- `template<typename _OffsetType, typename _ExtentType, typename _StrideType>`  
`constexpr bool std::__mdspan::__is_extent_slice< extent_slice< _OffsetType, _ExtentType, _StrideType > >`
- `template<typename _Tp>`  
`constexpr bool std::__mdspan::__is_extents`
- `template<typename _IndexType, size_t... _Extents>`  
`constexpr bool std::__mdspan::__is_extents< extents< _IndexType, _Extents... > >`
- `template<typename _Mapping>`  
`constexpr bool std::__mdspan::__is_left_padded_mapping`
- `template<typename _Mapping>`  
`constexpr bool std::__mdspan::__is_padded_mapping`
- `template<typename _Tp>`  
`constexpr bool std::__mdspan::__is_range_slice`
- `template<typename _FirstType, typename _LastType, typename _StrideType>`  
`constexpr bool std::__mdspan::__is_range_slice< range_slice< _FirstType, _LastType, _StrideType > >`
- `template<typename _Mapping>`  
`constexpr bool std::__mdspan::__is_right_padded_mapping`
- `template<typename _Tp>`  
`constexpr bool std::__is_submdspan_mapping_result`
- `template<typename _Mapping>`  
`constexpr bool std::__is_submdspan_mapping_result< submdspan_mapping_result< _Mapping > >`
- `template<array _Extents>`  
`constexpr auto std::__mdspan::__rev_partial_prods`
- `constexpr full_extent_t std::full_extent`

### 7.603.1 Detailed Description

This is a Standard C++ Library header.

## 7.604 experimental/memory File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define __cpp_lib_experimental_observer_ptr`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY`

### Functions

- `template<typename _Tp>`  
`constexpr observer_ptr< _Tp > std::experimental::make_observer (_Tp *__p) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`bool std::experimental::operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`

- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp>`  
`constexpr bool std::experimental::operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp>`  
`constexpr void std::experimental::swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`

### 7.604.1 Detailed Description

This is a TS C++ Library header.

## 7.605 ext/memory File Reference

### Classes

- class [`\_\_gnu\_cxx::Temporary\_buffer<\_ForwardIterator, \_Tp>`](#)
- struct [`\_\_gnu\_cxx::temporary\_buffer<\_ForwardIterator, \_Tp>`](#)

### Namespaces

- namespace [`\_\_gnu\_cxx`](#)

### Macros

- `#define _EXT_MEMORY`

### Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
[`std::pair<\_InputIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::\_\_uninitialized\_copy\_n`](#) (`_InputIter __first`, `_Size __count`, `↔`  
`_ForwardIter __result`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
[`std::pair<\_InputIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::\_\_uninitialized\_copy\_n`](#) (`_InputIter __first`, `_Size __count`, `↔`  
`_ForwardIter __result`, [`std::input\_iterator\_tag`](#))
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>`  
[`std::pair<\_RandomAccessIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::\_\_uninitialized\_copy\_n`](#) (`_RandomAccessIter __first`, `_Size __count`, `_ForwardIter __result`, [`std::random\_access\_iterator\_tag`](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>`  
[`std::pair<\_InputIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::\_\_uninitialized\_copy\_n\_a`](#) (`_InputIter __first`, `_Size __count`,  
`_ForwardIter __result`, `_Allocator __alloc`)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>`  
[`std::pair<\_InputIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::\_\_uninitialized\_copy\_n\_a`](#) (`_InputIter __first`, `_Size __count`,  
`_ForwardIter __result`, [`std::allocator<\_Tp>`](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter>`  
[`std::pair<\_InputIter, \_ForwardIter>`](#) [`\_\_gnu\_cxx::uninitialized\_copy\_n`](#) (`_InputIter __first`, `_Size __count`, `↔`  
`_ForwardIter __result`)

### 7.605.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.606 memory File Reference

### 7.606.1 Detailed Description

This is a Standard C++ Library header.

## 7.607 experimental/memory\_resource File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_memory_resources`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE`
- `#define _GLIBCXX_MAX_ALIGN_MATCHES_MALLOC`

### Typedefs

- `template<typename _Alloc>`  
using `std::experimental::fundamentals_v2::pmr::resource_adaptor`

### Functions

- `memory_resource * std::experimental::fundamentals\_v2::pmr::get\_default\_resource ()` noexcept
- `memory_resource * std::experimental::fundamentals\_v2::pmr::new\_delete\_resource ()` noexcept
- `memory_resource * std::experimental::fundamentals\_v2::pmr::null\_memory\_resource ()` noexcept
- `bool std::experimental::fundamentals\_v2::pmr::operator!= (const memory_resource &__a, const memory_resource &__b)` noexcept
- `template<class _Tp1, class _Tp2>`  
`bool std::experimental::fundamentals\_v2::pmr::operator!= (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b)` noexcept
- `bool std::experimental::fundamentals\_v2::pmr::operator== (const memory_resource &__a, const memory_resource &__b)` noexcept
- `template<class _Tp1, class _Tp2>`  
`bool std::experimental::fundamentals\_v2::pmr::operator== (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b)` noexcept
- `memory_resource * std::experimental::fundamentals\_v2::pmr::set\_default\_resource (memory_resource *__r)` noexcept

### 7.607.1 Detailed Description

This is a TS C++ Library header.

### 7.607.2 Function Documentation

#### **get\_default\_resource()**

```
memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () [inline],
[noexcept]
```

Get the current default resource.

#### **set\_default\_resource()**

```
memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (
 memory_resource * __r) [inline], [noexcept]
```

Change the default resource and return the previous one.

## 7.608 memory\_resource File Reference

### Classes

- class [std::pmr::monotonic\\_buffer\\_resource](#)
- struct [std::pmr::pool\\_options](#)
- class [std::pmr::unsynchronized\\_pool\\_resource](#)

### Namespaces

- namespace [std](#)

### Functions

- [memory\\_resource \\* std::pmr::get\\_default\\_resource \(\)](#) noexcept
- [memory\\_resource \\* std::pmr::new\\_delete\\_resource \(\)](#) noexcept
- [memory\\_resource \\* std::pmr::null\\_memory\\_resource \(\)](#) noexcept
- [memory\\_resource \\* std::pmr::set\\_default\\_resource \(memory\\_resource \\* \\_\\_r\)](#) noexcept

### 7.608.1 Detailed Description

This is a Standard C++ Library header.

This header declares the [pmr](#) (`std::pmr`) memory resources.

### 7.608.2 Function Documentation

#### **get\_default\_resource()**

```
memory_resource * std::pmr::get_default_resource () [noexcept]
```

Get the current default memory resource pointer.

#### **null\_memory\_resource()**

```
memory_resource * std::pmr::null_memory_resource () [nodiscard], [noexcept]
```

A `pmr::memory_resource` that always throws `bad_alloc`.

#### **set\_default\_resource()**

```
memory_resource * std::pmr::set_default_resource (
 memory_resource * __r) [noexcept]
```

Replace the default memory resource pointer.

## 7.609 mutex File Reference

### Classes

- struct [std::once\\_flag](#)
- class [std::recursive\\_mutex](#)
- class [std::recursive\\_timed\\_mutex](#)
- class [std::timed\\_mutex](#)

### Namespaces

- namespace [std](#)

### Functions

- void [std::\\_\\_once\\_proxy](#) (void)
- template<typename \_Callable, typename... \_Args>  
void [std::call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void [std::lock](#) (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)
- template<typename \_L1, typename \_L2, typename... \_L3>  
int [std::try\\_lock](#) (\_L1 &\_\_l1, \_L2 &\_\_l2, \_L3 &... \_\_l3)

#### 7.609.1 Detailed Description

This is a Standard C++ Library header.

## 7.610 numbers File Reference

#### 7.610.1 Detailed Description

This is a Standard C++ Library header.

## 7.611 experimental/numeric File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_gcd_lcm`
- `#define _GLIBCXX_EXPERIMENTAL_NUMERIC`

### Functions

- template<typename \_Mn, typename \_Nn>  
constexpr [common\\_type\\_t](#)<\_Mn, \_Nn> [std::experimental::gcd](#) (\_Mn \_\_m, \_Nn \_\_n) noexcept
- template<typename \_Mn, typename \_Nn>  
constexpr [common\\_type\\_t](#)<\_Mn, \_Nn> [std::experimental::lcm](#) (\_Mn \_\_m, \_Nn \_\_n)

#### 7.611.1 Detailed Description

This is a TS C++ Library header.

## 7.612 ext/numeric File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_NUMERIC`

### Functions

- `template<typename _Tp, typename _Integer>`  
`_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _ForwardIterator, typename _Tp>`  
`constexpr void \_\_gnu\_cxx::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _Tp, typename _Integer>`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation>`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`

#### 7.612.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 7.613 numeric File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Macros

- `#define _PSTL_NUMERIC_FORWARD_DECLARED`

### Typedefs

- `template<typename _Out, typename _Tp>`  
`using std::ranges::iota\_result`

### Functions

- `template<typename _Res, typename _Tp>`  
`constexpr _Res std::\_\_detail::\_\_abs\_r (_Tp __val)`
- `template<typename>`  
`void std::\_\_detail::\_\_abs\_r (bool)=delete`
- `template<typename _Tp>`  
`constexpr _Tp std::\_\_detail::\_\_gcd (_Tp __m, _Tp __n)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp>`  
`constexpr _OutputIterator std::exclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _Tp __init)`

- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _OutputIterator std::exclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::inclusive\_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _BinaryOperation __binary_op, _Tp __init)`
- `template<typename _InputIterator>`  
`constexpr iterator\_traits<_InputIterator>::value_type std::reduce (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator, typename _Tp>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation>`  
`constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform\_exclusive\_scan (_InputIterator __first, _InputIterator __last, _Output↵`  
`Iterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _OutputIterator std::transform\_inclusive\_scan (_InputIterator __first, _InputIterator __last, _Output↵`  
`Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>`  
`constexpr _OutputIterator std::transform\_inclusive\_scan (_InputIterator __first, _InputIterator __last, _Output↵`  
`Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>`  
`constexpr _Tp std::transform\_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation`  
`__binary_op, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp>`  
`constexpr _Tp std::transform\_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↵`  
`Operation2>`  
`constexpr _Tp std::transform\_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

## Variables

- `constexpr __iota_fn std::ranges::iota`

### 7.613.1 Detailed Description

This is a Standard C++ Library header.

## 7.614 parallel/numeric File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Functions

- `template<typename __RAIter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::__parallel::__accumulate_switch (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation`  
`__binary_op, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __↵`  
`binary_op, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator ↵`  
`__result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _Output↵`  
`Iterator __result, _BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename ↵`  
`IteratorTag1, typename _IteratorTag2>`  
`_Tp std::__parallel::__inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, ↵`  
`BinaryFunction1 __binary_op1, BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _Binary↵`  
`Function2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism=\_\_gnu\_parallel::parallel\_unbal`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`_BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`_BinaryOperation __bin_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation>`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`\_\_gnu\_parallel::sequential\_tag)`



- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`

### 7.614.1 Detailed Description

Parallel STL function calls corresponding to `stl_numeric.h`. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 7.615 experimental/optional File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::bad\\_optional\\_access](#)
- struct [std::hash< experimental::optional< \\_Tp > >](#)
- struct [std::experimental::fundamentals\\_v1::in\\_place\\_t](#)
- struct [std::experimental::fundamentals\\_v1::nullopt\\_t](#)
- class [std::experimental::fundamentals\\_v1::optional< \\_Tp >](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

## Macros

- #define `__cpp_lib_experimental_optional`
- #define `_GLIBCXX_EXPERIMENTAL_OPTIONAL`

## Variables

- constexpr `in_place_t` `std::experimental::in_place`
- constexpr `nullopt_t` `std::experimental::nullopt`

### 7.615.1 Detailed Description

This is a TS C++ Library header.

## 7.616 optional File Reference

### 7.616.1 Detailed Description

This is a Standard C++ Library header.

## 7.617 ostream File Reference

### Namespaces

- namespace `std`

### Functions

- template<typename `_CharT`, typename `_Traits`>  
`basic_ostream`< `_CharT`, `_Traits` > & `std::endl` (`basic_ostream`< `_CharT`, `_Traits` > & `__os`)
- template<typename `_CharT`, typename `_Traits`>  
`basic_ostream`< `_CharT`, `_Traits` > & `std::ends` (`basic_ostream`< `_CharT`, `_Traits` > & `__os`)
- template<typename `_CharT`, typename `_Traits`>  
`basic_ostream`< `_CharT`, `_Traits` > & `std::flush` (`basic_ostream`< `_CharT`, `_Traits` > & `__os`)

### 7.617.1 Detailed Description

This is a Standard C++ Library header.

## 7.618 algo.h File Reference

### Classes

- struct `std::__parallel::__CRandNumber`< `_MustBeInt` >

### Namespaces

- namespace `std`
- namespace `std::__parallel`

### Functions

- template<typename `_FIterator`, typename `_BinaryPredicate`, typename `_IteratorTag`>  
`_FIterator` `std::__parallel::__adjacent_find_switch` (`_FIterator` `__begin`, `_FIterator` `__end`, `_BinaryPredicate` `↔`  
`__pred`, `_IteratorTag`)

- `template<typename _FIterator, typename _IteratorTag>`  
`_FIterator std::parallel::adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate>`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter>`  
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter __begin, _Iter __end, const _Tp & __value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>`  
`_Iter std::parallel::find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>`  
`_RAIter std::parallel::find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag>`  
`_Iter std::parallel::find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag>`  
`_Iter std::parallel::find_switch (_Iter __begin, _Iter __end, const _Tp & __val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag>`  
`_Function std::parallel::for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function>`  
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>`  
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator>`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag>`  
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator>`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`

- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`  
`_OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::parallel::merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`  
`_OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag>`  
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare>`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag>`  
`_FIterator std::parallel::partition_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, __IteratorTag)`
- `template<typename _RAIter, typename _Predicate>`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>`  
`void std::parallel::replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp`  
`& __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp>`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value,`  
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag>`  
`void std::parallel::replace_switch (_FIterator __begin, _FIterator __end, const _Tp & __old_value, const`  
`_Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp>`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp`  
`& __new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>`  
`_FIterator std::parallel::search_n_switch (_FIterator __begin, _FIterator __end, _Integer __count, const`  
`_Tp & __val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_RAIter std::parallel::search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val,`  
`_BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>`  
`_FIterator1 std::parallel::search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2,`  
`_FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2>`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`  
`_IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`

- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2`  
`__begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1,`  
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>`  
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate>`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation`  
`__unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation`  
`__unary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism`  
`__parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator`  
`__result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3`  
`__result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, gnu\\_parallel::Parallelism \_\_parallelism\_tag\)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate`  
`__pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>`  
`_RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator`  
`__out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator std::parallel::adjacent_find (_Filterator __begin, _Filterator __end, gnu\_parallel::sequential\_tag)`

- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_↵`  
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate>`  
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`  
`value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate>`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`  
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp>`  
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator>`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, ↵`  
`_BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate>`  
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, ↵`  
`_BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate>`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Function>`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iterator, typename _Function>`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::Parallelism`  
`__parallelism_tag)`
- `template<typename _FIterator, typename _Generator>`  
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _Generator __gen)`

- `template<typename _Filterator, typename _Generator>`  
`void std::__parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism`  
`__parallelism_tag)`
- `template<typename _Filterator, typename _Generator>`  
`void std::__parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator>`  
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size __n, _Generator __gen,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism ↵`  
`__parallelism_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare>`  
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::Parallelism ↵`  
`__parallelism_tag)`
- `template<typename _Filterator>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _Filterator, typename _Compare>`  
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp,`  
`\_\_gnu\_parallel::Parallelism __parallelism_tag)`



- `template<typename _FIterator, typename _Compare>`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Predicate>`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _FIterator, typename _Predicate>`  
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator>`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Tp>`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Tp>`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp>`  
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp>`  
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::sequential_tag)`



- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2>`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _ForwardIterator, typename _Searcher>`  
`_ForwardIterator std::parallel::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher & \_\_searcher)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val)`
- `template<typename _FIterator, typename _Integer, typename _Tp>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator>`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __↵`  
`parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __↵`  
`parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag`  
`__parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __↵`  
`parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter>`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __↵`  
`parallelism)`
- `template<typename _RAIter>`  
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __↵`  
`parallelism)`

- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag ↔  
__parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism>`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↔  
parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↔  
Operation __unary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↔  
Operation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↔  
Operation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator  
__result, _BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator  
__result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation>`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator  
__result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out,  
\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate  
__pred)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate>`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate  
__pred, \_\_gnu\_parallel::sequential\_tag)`

### 7.618.1 Detailed Description

Parallel STL function calls corresponding to the `std::algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 7.619 `algo`.h File Reference

### Namespaces

- namespace `std`
- namespace `std::parallel`

### Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool std::parallel::equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::parallel::equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`bool std::parallel::lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>`  
`pair<_Iter1, _Iter2> std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair<_RAIter1, _RAIter2> std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate>`  
`pair<_RAIter1, _RAIter2> std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>`  
`_Filterator1 std::parallel::search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate>`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2>`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate>`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>`  
`constexpr _FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>`  
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`

### 7.619.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 7.620 `balanced_quicksort.h` File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, [\\_ThreadIndex](#) \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) ([\\_QSBThreadLocal<\\_RAIter>](#) \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, [\\_ThreadIndex](#) \_\_iam, [\\_ThreadIndex](#) \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare>  
[std::iterator\\_traits<\\_RAIter>::difference\\_type](#) [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, [\\_ThreadIndex](#) \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping](#) ([\\_QSBThreadLocal<\\_RAIter>](#) \*\_\_tls, \_Compare &\_\_comp, [\\_ThreadIndex](#) \_\_iam, bool \_\_wait)

#### 7.620.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

## 7.621 `base.h` File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate<\\_Predicate, argument\\_type>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess<\\_T1, \\_T2, \\_Compare>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Less<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies<\\_Tp1, \\_Tp2, \\_Result>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus<\\_Tp1, \\_Tp2, \\_Result>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence<\\_Tp, \\_DifferenceTp>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator<\\_Tp, \\_DifferenceTp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)
- namespace [\\_\\_gnu\\_sequential](#)
- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

**Macros**

- `#define _GLIBCXX_PARALLEL_ASSERT(_Condition)`

**Functions**

- `void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)`
- `_CASable __gnu_parallel::__encode2 (int __a, int __b)`
- `_ThreadIndex __gnu_parallel::__get_max_threads ()`
- `bool __gnu_parallel::__is_parallel (const _Parallelism __p)`
- `template<typename _RAIter, typename _Compare>  
_RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _Size>  
_Size __gnu_parallel::__rd_log2 (_Size __n)`
- `template<typename _Tp>  
const _Tp & __gnu_parallel::__max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp>  
const _Tp & __gnu_parallel::__min (const _Tp &__a, const _Tp &__b)`

**7.621.1 Detailed Description**

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

**7.622 basic\_iterator.h File Reference****7.622.1 Detailed Description**

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

**7.623 checkers.h File Reference****Namespaces**

- namespace `__gnu_parallel`

**Functions**

- `template<typename _Iter, typename _Compare>  
bool __gnu_parallel::__is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`

**7.623.1 Detailed Description**

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

**7.624 compiletime\_settings.h File Reference****Macros**

- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_PARALLEL_ASSERTIONS`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

### 7.624.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### 7.624.2 Macro Definition Documentation

#### **`_GLIBCXX_CALL`**

```
#define _GLIBCXX_CALL(
 __n)
```

Macro to produce log message when entering a function.

#### **Parameters**

|                  |             |
|------------------|-------------|
| <code>__n</code> | Input size. |
|------------------|-------------|

#### **See also**

`_GLIBCXX_VERBOSE_LEVEL`

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__find_template()`, `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_merge()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_trees()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_parallel::__parallel_sort_mwms()`.

#### **`_GLIBCXX_PARALLEL_ASSERTIONS`**

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

#### **`_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`**

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the L1 cache for `gnu_parallel::__parallel_random_shuffle()`.

#### **`_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`**

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

#### **`_GLIBCXX_SCALE_DOWN_FPU`**

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.



**`_GLIBCXX_VERBOSE_LEVEL`**

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

**7.625 `equally_split.h` File Reference****Namespaces**

- namespace `__gnu_parallel`

**Functions**

- `template<typename _DifferenceType, typename _OutputIterator>`  
`_OutputIterator __gnu_parallel::__equally_split` (`_DifferenceType __n`, `_ThreadIndex __num_threads`, `_OutputIterator __s`)
- `template<typename _DifferenceType>`  
`_DifferenceType __gnu_parallel::__equally_split_point` (`_DifferenceType __n`, `_ThreadIndex __num_threads`, `_ThreadIndex __thread_no`)

**7.625.1 Detailed Description**

This file is a GNU parallel extension to the Standard C++ Library.

**7.626 `features.h` File Reference****Macros**

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`
- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

**7.626.1 Detailed Description**

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

**7.626.2 Macro Definition Documentation****`_GLIBCXX_BAL_QUICKSORT`**

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

**\_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS**

#define \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS  
Include the equal-sized blocks variant for std::find.

See also

\_\_gnu\_parallel::\_Settings::find\_algorithm

**\_GLIBCXX\_FIND\_EQUAL\_SPLIT**

#define \_GLIBCXX\_FIND\_EQUAL\_SPLIT  
Include the equal splitting variant for std::find.

See also

\_\_gnu\_parallel::\_Settings::find\_algorithm

**\_GLIBCXX\_FIND\_GROWING\_BLOCKS**

#define \_GLIBCXX\_FIND\_GROWING\_BLOCKS  
Include the growing blocks variant for std::find.

See also

\_\_gnu\_parallel::\_Settings::find\_algorithm

**\_GLIBCXX\_MERGESORT**

#define \_GLIBCXX\_MERGESORT  
Include parallel multi-way mergesort.

See also

\_\_gnu\_parallel::\_Settings::sort\_algorithm

**\_GLIBCXX\_QUICKSORT**

#define \_GLIBCXX\_QUICKSORT  
Include parallel unbalanced quicksort.

See also

\_\_gnu\_parallel::\_Settings::sort\_algorithm

**\_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

#define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING  
Include the dynamic balancing variant for \_Rb\_tree::insert\_unique(\_Iter beg, \_Iter \_\_end).

See also

\_\_gnu\_parallel::\_Rb\_tree

**\_GLIBCXX\_TREE\_FULL\_COPY**

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

**\_GLIBCXX\_TREE\_INITIAL\_SPLITTING**

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

**7.627 find.h File Reference****Namespaces**

- namespace [\\_\\_gnu\\_parallel](#)

**Functions**

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [constant\\_size\\_blocks\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [equal\\_split\\_tag](#))
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>`  
[std::pair](#)< [\\_RAIter1](#), [\\_RAIter2](#) > [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [growing\\_blocks\\_tag](#))

**7.627.1 Detailed Description**

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

**7.628 find\_selectors.h File Reference****Classes**

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector](#)< [\\_FIterator](#) >
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)

**Namespaces**

- namespace [\\_\\_gnu\\_parallel](#)

### 7.628.1 Detailed Description

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## 7.629 `for_each.h` File Reference

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>  
_UserOp __gnu_parallel::__for_each_template_random_access (_Iter __begin, _Iter __end, _UserOp __user←  
_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename  
std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __parallelism_tag)`

### 7.629.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like `workstealing.h`, `par_loop.h`, `omp_loop.h`, and `omp_loop_←  
static.h`. This file is a GNU parallel extension to the Standard C++ Library.

## 7.630 `for_each_selectors.h` File Reference

### Classes

- `struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >`
- `struct __gnu_parallel::__accumulate_selector< _It >`
- `struct __gnu_parallel::__adjacent_difference_selector< _It >`
- `struct __gnu_parallel::__count_if_selector< _It, _Diff >`
- `struct __gnu_parallel::__count_selector< _It, _Diff >`
- `struct __gnu_parallel::__fill_selector< _It >`
- `struct __gnu_parallel::__for_each_selector< _It >`
- `struct __gnu_parallel::__generate_selector< _It >`
- `struct __gnu_parallel::__generic_for_each_selector< _It >`
- `struct __gnu_parallel::__identity_selector< _It >`
- `struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`
- `struct __gnu_parallel::__max_element_reduct< _Compare, _It >`
- `struct __gnu_parallel::__min_element_reduct< _Compare, _It >`
- `struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`
- `struct __gnu_parallel::__replace_selector< _It, _Tp >`
- `struct __gnu_parallel::__transform1_selector< _It >`
- `struct __gnu_parallel::__transform2_selector< _It >`
- `struct __gnu_parallel::__DummyReduct`
- `struct __gnu_parallel::__Nothing`

### Namespaces

- namespace `__gnu_parallel`

### 7.630.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

## 7.631 iterator.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair<\\_Iterator1, \\_Iterator2, \\_IteratorCategory>](#)
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple<\\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

#### 7.631.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## 7.632 list\_partition.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter>`  
void [\\_\\_gnu\\_parallel::\\_shrink](#) (std::vector<\_Iter> &\_\_os\_starts, size\_t &\_\_count\_to\_two, size\_t &\_\_range\_length)
- `template<typename _Iter>`  
void [\\_\\_gnu\\_parallel::\\_shrink\\_and\\_double](#) (std::vector<\_Iter> &\_\_os\_starts, size\_t &\_\_count\_to\_two, size\_t &\_\_range\_length, const bool \_\_make\_twice)
- `template<typename _Iter, typename _FuncType>`  
size\_t [\\_\\_gnu\\_parallel::list\\_partition](#) (const \_Iter \_\_begin, const \_Iter \_\_end, \_Iter \*\_\_starts, size\_t \*\_\_lengths, const int \_\_num\_parts, \_FuncType &\_\_f, int \_\_oversampling=0)

#### 7.632.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 7.633 losertree.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>::\\_Loser](#)
- struct [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase<\\_Tp, \\_Compare>::\\_Loser](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTree<\\_\\_stable, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTree<false, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeBase<\\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer<\\_\\_stable, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointer<false, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase<\\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded<\\_\\_stable, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded<false, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase<\\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded<\\_\\_stable, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded<false, \\_Tp, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase<\\_Tp, \\_Compare>](#)

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 7.633.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

## 7.634 merge.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator\_traits<_RAIter1>::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>  
_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator\_traits<_RAIter1>::difference_type __max_length, _Compare __comp)`

### 7.634.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

## 7.635 multiseq\_selection.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_\\_Lexicographic](#)<\_T1, \_T2, \_Compare >
- class [\\_\\_gnu\\_parallel::\\_\\_LexicographicReverse](#)<\_T1, \_T2, \_Compare >

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_\_S(__i)`
- `#define \_\_S(__i)`

## Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 7.635.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171-177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

## 7.636 multiway\_merge.h File Reference

### Classes

- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, __RAIter3, __DifferenceTp, _Compare >`
- `struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, __DifferenceTp, _Compare >`
- `class __gnu_parallel::__GuardedIterator< _RAIter, _Compare >`
- `struct __gnu_parallel::__LoserTreeTraits< _Tp >`
- `struct __gnu_parallel::__SamplingSorter< __stable, _RAIter, __StrictWeakOrdering >`
- `struct __gnu_parallel::__SamplingSorter< false, _RAIter, __StrictWeakOrdering >`

### Namespaces

- namespace `__gnu_parallel`

### Macros

- `#define _GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`
- `#define _GLIBCXX_PARALLEL_LENGTH(__s)`
- `#define _GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- `#define _GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>  
_OutputIterator gnu\_parallel::merge\_advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>  
void gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>  
_RAIter3 gnu\_parallel::multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>  
void gnu\_parallel::multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>  
_RAIterOut gnu\_parallel::multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`



- [illegible]

### 7.636.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

### 7.636.2 Macro Definition Documentation

#### **`_GLIBCXX_PARALLEL_LENGTH`**

```
#define _GLIBCXX_PARALLEL_LENGTH(
 __s)
```

Length of a sequence described by a pair of iterators.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_sequential\\_multiway\\_merge\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_exact\\_splitting\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_loser\\_tree\(\)](#), [\\_\\_gnu\\_parallel::multiway\\_merge\\_sampling\\_splitting\(\)](#), and [\\_\\_gnu\\_parallel::parallel\\_multiway\\_merge\(\)](#).

## 7.637 `multiway_mergesort.h` File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::Piece<\\_DifferenceTp>](#)
- struct [\\_\\_gnu\\_parallel::PMWSSortingData<\\_RAIter>](#)
- struct [\\_\\_gnu\\_parallel::SplitConsistently<\\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)
- struct [\\_\\_gnu\\_parallel::SplitConsistently<false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)
- struct [\\_\\_gnu\\_parallel::SplitConsistently<true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_DifferenceTp>  
void [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples](#) ([\\_PMWSSortingData<\\_RAIter>](#) \*\_\_sd, [\\_DifferenceTp](#) \_\_num←  
\_samples)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Compare](#) \_\_comp, [\\_ThreadIndex](#)  
\_\_num\_threads)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare>  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu](#) ([\\_PMWSSortingData<\\_RAIter>](#) \*\_\_sd, [\\_Compare](#) &\_\_comp)

### 7.637.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

## 7.638 `numericfwd.h` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Functions

- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag>`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _Iter, typename _Tp, typename _Tag>`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>`  
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism=__gnu_parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>`  
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _Tp>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper>`  
`_Tp std::__parallel::__accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::__adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::__inner_product (_Iter1, _Iter1, _Iter2, _Tp)`

- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter>`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper>`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`

### 7.638.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 7.639 omp\_loop.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits<_RAIter>::difference_type __bound)`

### 7.639.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

## 7.640 omp\_loop\_static.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>`  
`_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop\_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits<_RAIter>::difference_type __bound)`

### 7.640.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

## 7.641 par\_loop.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu  
& __f, _Red __r, _Result __base, _Result & __output, typename std::iterator\_traits< _RAIter >::difference_type  
__bound)`

### 7.641.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## 7.642 parallel.h File Reference

### 7.642.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

## 7.643 partial\_sum.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result,  $\leftrightarrow$   
_BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator  
__result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __ $\leftrightarrow$   
result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::difference_type __n)`

### 7.643.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## 7.644 partition.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define` [\\_GLIBCXX\\_VOLATILE](#)

### Functions

- `template<typename _RAIter, typename _Compare>`  
`void` [\\_\\_gnu\\_parallel::\\_\\_parallel\\_nth\\_element](#) (`_RAIter` \_\_begin, `_RAIter` \_\_nth, `_RAIter` \_\_end, `_Compare` \_\_comp)
- `template<typename _RAIter, typename _Compare>`  
`void` [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partial\\_sort](#) (`_RAIter` \_\_begin, `_RAIter` \_\_middle, `_RAIter` \_\_end, `_Compare` \_\_comp)
- `template<typename _RAIter, typename _Predicate>`  
`std::iterator_traits<_RAIter>::difference_type` [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition](#) (`_RAIter` \_\_begin, `_RAIter` \_\_end, `_Predicate` \_\_pred, `_ThreadIndex` \_\_num\_threads)

#### 7.644.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

#### 7.644.2 Macro Definition Documentation

##### `_GLIBCXX_VOLATILE`

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.

Referenced by [\\_\\_gnu\\_parallel::\\_\\_parallel\\_partition\(\)](#).

## 7.645 queue.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define` [\\_GLIBCXX\\_VOLATILE](#)

#### 7.645.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### 7.645.2 Macro Definition Documentation

#### `_GLIBCXX_VOLATILE`

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

## 7.646 quicksort.h File Reference

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare>`  
`std::iterator\_traits<\_RAIter>::difference\_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, \_RAIter __end, \_Compare __comp, std::iterator\_traits<\_RAIter>::difference\_type __pivot_rank, std::iterator\_traits<\_RAIter>::difference\_type __num_samples, \_ThreadIndex __num_threads)`

### 7.646.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

## 7.647 random\_number.h File Reference

### Classes

- class `__gnu_parallel::_RandomNumber`

### Namespaces

- namespace `__gnu_parallel`

### 7.647.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

## 7.648 random\_shuffle.h File Reference

### Classes

- struct `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>`
- struct `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>`

### Namespaces

- namespace `__gnu_parallel`

## Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::BinIndex](#)

## Functions

- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator \_\_rng=[\\_RandomNumber](#)())
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, typename [std::iterator\\_traits](#)<\_RAIter>::difference\_type \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, \_RandomNumberGenerator &\_\_rng)
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu](#) ([\\_DRSSorterPU](#)<\_RAIter, \_RandomNumberGenerator> \* \_\_pus)
- template<typename \_RandomNumberGenerator>  
int [\\_\\_gnu\\_parallel::\\_\\_random\\_number\\_pow2](#) (int \_\_logp, \_RandomNumberGenerator &\_\_rng)
- template<typename \_Tp>  
[\\_Tp](#) [\\_\\_gnu\\_parallel::\\_\\_round\\_up\\_to\\_pow2](#) (\_Tp \_\_x)
- template<typename \_RAIter, typename \_RandomNumberGenerator>  
void [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rng)

### 7.648.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

## 7.649 search.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_DifferenceTp>  
void [\\_\\_gnu\\_parallel::\\_\\_calc\\_borders](#) (\_RAIter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \* \_\_off)
- template<typename \_\_RAIter1, typename \_\_RAIter2, typename \_Pred>  
[\\_\\_RAIter1](#) [\\_\\_gnu\\_parallel::\\_\\_search\\_template](#) (\_\_RAIter1 \_\_begin1, \_\_RAIter1 \_\_end1, \_\_RAIter2 \_\_begin2, [\\_\\_RAIter2](#) \_\_end2, \_Pred \_\_pred)

### 7.649.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## 7.650 set\_operations.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)



## Functions

- `template<typename _Iter, typename _OutputIterator>`  
`_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, ↵`  
`_OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, ↵`  
`_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2,`  
`_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation>`  
`_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, ↵`  
`_Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter`  
`__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare>`  
`_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter`  
`__end2, _OutputIterator __result, _Compare __comp)`

### 7.650.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 7.651 settings.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Settings](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_PARALLEL\_CONDITION(__c)`

### 7.651.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

### 7.651.2 Deciding whether to run an algorithm in parallel.

There are several ways the user can switch on and off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::↵`  
`sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

- a. `force_sequential`, meaning the sequential algorithm is executed.
- b. `force_parallel`, meaning the parallel algorithm is executed.
- c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

### 7.651.3 Macro Definition Documentation

#### `_GLIBCXX_PARALLEL_CONDITION`

```
#define _GLIBCXX_PARALLEL_CONDITION(
 __c)
```

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

#### Parameters

|                  |                                                                                                                                                                       |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__c</code> | A condition that is convertible to bool that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> . Usually a decision based on the input size. |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 7.652 `sort.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename \_Parallelism>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel\_tag __↔`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare>`  
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __↔`  
`parallelism)`

### 7.652.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

## 7.653 tags.h File Reference

### Classes

- `struct \_\_gnu\_parallel::balanced\_quicksort\_tag`
- `struct \_\_gnu\_parallel::balanced\_tag`
- `struct \_\_gnu\_parallel::constant\_size\_blocks\_tag`
- `struct \_\_gnu\_parallel::default\_parallel\_tag`
- `struct \_\_gnu\_parallel::equal\_split\_tag`
- `struct \_\_gnu\_parallel::exact\_tag`
- `struct \_\_gnu\_parallel::find\_tag`
- `struct \_\_gnu\_parallel::growing\_blocks\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag`
- `struct \_\_gnu\_parallel::multiway\_mergesort\_tag`
- `struct \_\_gnu\_parallel::omp\_loop\_static\_tag`
- `struct \_\_gnu\_parallel::omp\_loop\_tag`
- `struct \_\_gnu\_parallel::parallel\_tag`
- `struct \_\_gnu\_parallel::quicksort\_tag`
- `struct \_\_gnu\_parallel::sampling\_tag`
- `struct \_\_gnu\_parallel::sequential\_tag`
- `struct \_\_gnu\_parallel::unbalanced\_tag`

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 7.653.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## 7.654 types.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Typedefs

- `typedef int64_t \_\_gnu\_parallel::\_CASable`
- `typedef uint64_t \_\_gnu\_parallel::\_SequenceIndex`
- `typedef uint16_t \_\_gnu\_parallel::\_ThreadIndex`

## Enumerations

- enum [\\_\\_gnu\\_parallel::AlgorithmStrategy](#) { **heuristic** , **force\_sequential** , **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::FindAlgorithm](#) { **GROWING\_BLOCKS** , **CONSTANT\_SIZE\_BLOCKS** , **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::Parallelism](#) {  
    [\\_\\_gnu\\_parallel::sequential](#) , [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#) , [\\_\\_gnu\\_parallel::parallel\\_balanced](#) ,  
    [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop](#) ,  
    [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#) , [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::PartialSumAlgorithm](#) { **RECURSIVE** , **LINEAR** }
- enum [\\_\\_gnu\\_parallel::SortAlgorithm](#) { **MWMS** , **QS** , **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::SplittingAlgorithm](#) { **SAMPLING** , **EXACT** }

## Variables

- static const int [\\_\\_gnu\\_parallel::\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_CASable\\_mask](#)

### 7.654.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## 7.655 [unique\\_copy.h](#) File Reference

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename [\\_Iter](#), class [\\_OutputIterator](#)>  
    [\\_OutputIterator](#) [\\_\\_gnu\\_parallel::parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result)
- template<typename [\\_Iter](#), class [\\_OutputIterator](#), class [\\_BinaryPredicate](#)>  
    [\\_OutputIterator](#) [\\_\\_gnu\\_parallel::parallel\\_unique\\_copy](#) ([\\_Iter](#) \_\_first, [\\_Iter](#) \_\_last, [\\_OutputIterator](#) \_\_result, [\\_BinaryPredicate](#) \_\_binary\_pred)

### 7.655.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

## 7.656 [workstealing.h](#) File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Job<\\_DifferenceTp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Macros

- `#define \_GLIBCXX\_JOB\_VOLATILE`

**Functions**

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
_Op gnu\_parallel::for\_each\_template\_random\_access\_workstealing (_RAIter __begin, _RAIter __end, _Op  
__op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::↵  
difference_type __bound)`

**7.656.1 Detailed Description**

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720-748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

**7.657 [print File Reference](#)****7.657.1 Detailed Description**

This is a Standard C++ Library header.

**7.658 [queue File Reference](#)****7.658.1 Detailed Description**

This is a Standard C++ Library header.

**7.659 [experimental/random File Reference](#)****Namespaces**

- namespace [std](#)
- namespace [std::experimental](#)

**Macros**

- `#define \_\_cpp\_lib\_experimental\_randint`
- `#define \_GLIBCXX\_EXPERIMENTAL\_RANDOM`

**Functions**

- `std::default_random_engine & std::experimental::S\_randint\_engine ()`
- `template<typename _IntType>  
_IntType std::experimental::randint (_IntType __a, _IntType __b)`
- `void std::experimental::reseed ()`
- `void std::experimental::reseed (default\_random\_engine::result\_type __value)`

**7.659.1 Detailed Description**

This is a TS C++ Library header.

**7.660 [random File Reference](#)****7.660.1 Detailed Description**

This is a Standard C++ Library header.

## 7.661 ranges File Reference

### Classes

- class [std::ranges::empty\\_view<\\_Tp>](#)
- class [std::ranges::single\\_view<\\_Tp>](#)

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Fn, typename... _Iters>`  
using **std::ranges::\_\_detail::\_\_func\_handle\_t**
- `template<typename _Wp>`  
using **std::ranges::\_\_detail::\_\_iota\_diff\_t**
- `template<bool _Const, typename _Tp>`  
using **std::ranges::\_\_detail::\_\_maybe\_const\_t**
- `template<bool _Const, typename _Tp>`  
using **std::ranges::\_\_maybe\_const\_t**
- `template<bool _Present, typename _Tp, int _Disc = 0>`  
using **std::ranges::\_\_detail::\_\_maybe\_present\_t**
- `template<viewable_range _Range>`  
using **std::ranges::views::all\_t**
- `template<typename _Val>`  
using **std::ranges::istream\_view**
- `template<typename _Range>`  
using **std::ranges::keys\_view**
- `template<typename _Range>`  
using **std::ranges::values\_view**
- `template<typename _Val>`  
using **std::ranges::wistream\_view**

### Functions

- `template<typename _Tp>`  
constexpr **std::ranges::\_\_detail::\_\_as\_lvalue** (\_Tp &&\_\_t)
- `template<typename _Range>`  
constexpr auto **std::ranges::views::\_\_detail::\_\_drop\_of\_repeat\_view** (\_Range &&, range\_difference\_t<\_↵  
Range >)
- `template<typename _Tp, typename _Up>`  
void **std::ranges::views::\_\_adaptor::\_\_is\_range\_adaptor\_closure\_fn** (const \_Tp &, const \_RangeAdaptor↵  
Closure<\_Up > &)
- `template<typename _Fn, typename... _Iters>`  
constexpr auto **std::ranges::\_\_detail::\_\_func\_handle::\_\_select** ()
- `template<typename _Range>`  
constexpr auto **std::ranges::views::\_\_detail::\_\_take\_of\_repeat\_view** (\_Range &&, range\_difference\_t<\_↵  
Range >)
- `template<typename _Wp>`  
constexpr auto **std::ranges::\_\_detail::\_\_to\_signed\_like** (\_Wp \_\_w) noexcept
- `template<typename _Range>`  
**std::ranges::common\_view** (\_Range &&) -> common\_view< views::all\_t<\_Range > >

- `template<typename _Range>`  
`std::ranges::drop_view (_Range &&, range_difference_t< _Range >) -> drop_view< views::all_t< _Range >`  
`>`
- `template<typename _Range, typename _Pred>`  
`std::ranges::drop_while_view (_Range &&, _Pred) -> drop_while_view< views::all_t< _Range >, _Pred >`
- `template<typename _Range, typename _Pred>`  
`std::ranges::filter_view (_Range &&, _Pred) -> filter_view< views::all_t< _Range >, _Pred >`
- `template<typename _Winc, typename _Bound>`  
`std::ranges::iota_view (_Winc, _Bound) -> iota_view< _Winc, _Bound >`
- `template<typename _Range>`  
`std::ranges::join_view (_Range &&) -> join_view< views::all_t< _Range > >`
- `template<typename _Range, typename _Pattern>`  
`std::ranges::lazy_split_view (_Range &&, _Pattern &&) -> lazy_split_view< views::all_t< _Range >, views::all_t< _Pattern > >`
- `template<input_range _Range>`  
`std::ranges::lazy_split_view (_Range &&, range_value_t< _Range >) -> lazy_split_view< views::all_t< _Range >, single_view< range_value_t< _Range > > >`
- `template<typename _Lhs, typename _Rhs>`  
`constexpr auto std::ranges::views::__adaptor::operator| (_Lhs && _lhs, _Rhs && _rhs)`
- `template<typename _Self, typename _Range>`  
`constexpr auto std::ranges::views::__adaptor::operator| (_Range && _r, _Self && _self)`
- `template<typename _Range>`  
`std::ranges::ref_view (_Range &) -> ref_view< _Range >`
- `template<typename _Range>`  
`std::ranges::reverse_view (_Range &&) -> reverse_view< views::all_t< _Range > >`
- `template<typename _Tp>`  
`std::ranges::single_view (_Tp) -> single_view< _Tp >`
- `template<typename _Range, typename _Pattern>`  
`std::ranges::split_view (_Range &&, _Pattern &&) -> split_view< views::all_t< _Range >, views::all_t< _Pattern > >`
- `template<forward_range _Range>`  
`std::ranges::split_view (_Range &&, range_value_t< _Range >) -> split_view< views::all_t< _Range >, single_view< range_value_t< _Range > > >`
- `template<typename _Range>`  
`std::ranges::take_view (_Range &&, range_difference_t< _Range >) -> take_view< views::all_t< _Range > >`
- `template<typename _Range, typename _Pred>`  
`std::ranges::take_while_view (_Range &&, _Pred) -> take_while_view< views::all_t< _Range >, _Pred >`
- `template<typename _Range, typename _Fp>`  
`std::ranges::transform_view (_Range &&, _Fp) -> transform_view< views::all_t< _Range >, _Fp >`

## Variables

- `template<typename _Range>`  
`constexpr bool std::ranges::views::__detail::__is_basic_string_view`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::ranges::views::__detail::__is_basic_string_view< basic_string_view< _CharT, _Traits > >`
- `template<typename _Range>`  
`constexpr bool std::ranges::views::__detail::__is_empty_view`
- `template<typename _Tp>`  
`constexpr bool std::ranges::views::__detail::__is_empty_view< empty_view< _Tp > >`

- `template<typename _Range>`  
`constexpr bool std::ranges::views::__detail::__is_iota_view`
- `template<typename _Winc, typename _Bound>`  
`constexpr bool std::ranges::views::__detail::__is_iota_view< iota_view< _Winc, _Bound > >`
- `template<typename _Range>`  
`constexpr bool std::ranges::views::__detail::__is_repeat_view`
- `template<typename>`  
`constexpr bool std::ranges::views::__detail::__is_reverse_view`
- `template<typename _Vp>`  
`constexpr bool std::ranges::views::__detail::__is_reverse_view< reverse_view< _Vp > >`
- `template<typename>`  
`constexpr bool std::ranges::views::__detail::__is_reversible_subrange`
- `template<typename _Iter, subrange_kind _Kind>`  
`constexpr bool std::ranges::views::__detail::__is_reversible_subrange< subrange< reverse_iterator< ↵  
_Iter >, reverse_iterator< _Iter >, _Kind > >`
- `template<typename _Range>`  
`constexpr bool std::ranges::views::__detail::__is_subrange`
- `constexpr _All std::ranges::views::all`
- `constexpr _Common std::ranges::views::common`
- `constexpr _Counted std::ranges::views::counted`
- `constexpr _Drop std::ranges::views::drop`
- `constexpr _DropWhile std::ranges::views::drop_while`
- `template<size_t _Nm>`  
`constexpr _Elements< _Nm > std::ranges::views::elements`
- `template<typename _Tp>`  
`constexpr empty_view< _Tp > std::ranges::views::empty`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< common_view< _Tp > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< drop_view< _Tp > >`
- `template<typename _Tp, typename _Pred>`  
`constexpr bool std::ranges::enable_borrowed_range< drop_while_view< _Tp, _Pred > >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr bool std::ranges::enable_borrowed_range< elements_view< _Tp, _Nm > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< empty_view< _Tp > >`
- `template<typename _Winc, typename _Bound>`  
`constexpr bool std::ranges::enable_borrowed_range< iota_view< _Winc, _Bound > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< owning_view< _Tp > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< ref_view< _Tp > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< reverse_view< _Tp > >`
- `template<typename _Tp>`  
`constexpr bool std::ranges::enable_borrowed_range< take_view< _Tp > >`
- `constexpr _Filter std::ranges::views::filter`
- `constexpr _Iota std::ranges::views::iota`
- `template<typename _Tp>`  
`constexpr _Istream< _Tp > std::ranges::views::istream`
- `constexpr _Join std::ranges::views::join`
- `constexpr auto std::ranges::views::keys`



- constexpr `_LazySplit` **std::ranges::views::lazy\_split**
- constexpr `_Reverse` **std::ranges::views::reverse**
- constexpr `_Single` **std::ranges::views::single**
- constexpr `_Split` **std::ranges::views::split**
- constexpr `_Take` **std::ranges::views::take**
- constexpr `_TakeWhile` **std::ranges::views::take\_while**
- constexpr `_Transform` **std::ranges::views::transform**
- constexpr auto **std::ranges::views::values**

### 7.661.1 Detailed Description

This is a Standard C++ Library header.

## 7.662 experimental/ratio File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_RATIO`

### Variables

- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_equal\_v**
- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_greater\_equal\_v**
- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_greater\_v**
- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_less\_equal\_v**
- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_less\_v**
- `template<typename _R1, typename _R2>`  
constexpr bool **std::experimental::ratio\_not\_equal\_v**

### 7.662.1 Detailed Description

This is a TS C++ Library header.

## 7.663 ratio File Reference

### Classes

- struct [std::ratio<\\_Num, \\_Den>](#)
- struct [std::ratio\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_greater<\\_R1, \\_R2>](#)
- struct [std::ratio\\_greater\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_less<\\_R1, \\_R2>](#)
- struct [std::ratio\\_less\\_equal<\\_R1, \\_R2>](#)
- struct [std::ratio\\_not\\_equal<\\_R1, \\_R2>](#)

## Namespaces

- namespace [std](#)

## Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**
- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**
- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_add](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_divide](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_multiply](#)
- template<typename \_R1, typename \_R2>  
using [std::ratio\\_subtract](#)
- using **std::tera**

## Variables

- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_greater\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_equal\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_less\_v**
- template<typename \_R1, typename \_R2>  
constexpr bool **std::ratio\_not\_equal\_v**

### 7.663.1 Detailed Description

This is a Standard C++ Library header.

## 7.664 tr2/ratio File Reference

### Namespaces

- namespace `std`
- namespace `std::tr2`

#### 7.664.1 Detailed Description

This is a TR2 C++ Library header.

## 7.665 experimental/regex File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_REGEX`

### Typedefs

- `typedef match_results< const char * > std::experimental::fundamentals_v2::pmr::cmatch`
- `template<typename _BidirectionalIterator>`  
`using std::experimental::fundamentals_v2::pmr::match_results`
- `typedef match_results< string::const_iterator > std::experimental::fundamentals_v2::pmr::smatch`
- `typedef match_results< const wchar_t * > std::experimental::fundamentals_v2::pmr::wcmatch`
- `typedef match_results< wstring::const_iterator > std::experimental::fundamentals_v2::pmr::wsmatch`

#### 7.665.1 Detailed Description

This is a TS C++ Library header.

## 7.666 regex File Reference

### Namespaces

- namespace `std`

### Typedefs

- `using std::pmr::cmatch`
- `template<typename _BidirectionalIterator>`  
`using std::pmr::match_results`
- `using std::pmr::smatch`
- `using std::pmr::wcmatch`
- `using std::pmr::wsmatch`

#### 7.666.1 Detailed Description

This is a Standard C++ Library header.

## 7.667 `scoped_allocator` File Reference

### Classes

- class `std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >`

### Namespaces

- namespace `std`

#### 7.667.1 Detailed Description

This is a Standard C++ Library header.

## 7.668 `semaphore` File Reference

#### 7.668.1 Detailed Description

This is a Standard C++ Library header.

## 7.669 `debug/set` File Reference

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_SET`

#### 7.669.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.670 `experimental/set` File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_SET`

### Typedefs

- `template<typename _Key, typename _Compare = less<_Key>>>`  
  using `std::experimental::fundamentals_v2::pmr::multiset`
- `template<typename _Key, typename _Compare = less<_Key>>>`  
  using `std::experimental::fundamentals_v2::pmr::set`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`

### 7.670.1 Detailed Description

This is a TS C++ Library header.

## 7.671 set File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Key, typename _Cmp = std::less<_Key>>`  
`using std::pmr::multiset`
- `template<typename _Key, typename _Cmp = std::less<_Key>>`  
`using std::pmr::set`

### 7.671.1 Detailed Description

This is a Standard C++ Library header.

## 7.672 shared\_mutex File Reference

### Classes

- class [std::shared\\_lock< \\_Mutex >](#)
- class [std::shared\\_timed\\_mutex](#)

### Namespaces

- namespace [std](#)

### 7.672.1 Detailed Description

This is a Standard C++ Library header.

## 7.673 source\_location File Reference

### 7.673.1 Detailed Description

This is a Standard C++ Library header.

## 7.674 span File Reference

### 7.674.1 Detailed Description

This is a Standard C++ Library header.

## 7.675 sstream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#)

### Namespaces

- namespace [std](#)

### Macros

- `#define _GLIBCXX_LVAL_REF_QUAL`
- `#define _GLIBCXX_SSTREAM_ALWAYS_INLINE`

### Functions

- `template<class _CharT, class _Traits, class _Allocator>  
void std::swap (basic\_istream<\_CharT, \_Traits, \_Allocator> &__x, basic\_istream<\_CharT, \_Traits, \_Allocator> &__y)`
- `template<class _CharT, class _Traits, class _Allocator>  
void std::swap (basic\_ostringstream<\_CharT, \_Traits, \_Allocator> &__x, basic\_ostringstream<\_CharT, \_Traits, \_Allocator> &__y)`
- `template<class _CharT, class _Traits, class _Allocator>  
void std::swap (basic\_stringbuf<\_CharT, \_Traits, \_Allocator> &__x, basic\_stringbuf<\_CharT, \_Traits, \_Allocator> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _CharT, class _Traits, class _Allocator>  
void std::swap (basic\_stringstream<\_CharT, \_Traits, \_Allocator> &__x, basic\_stringstream<\_CharT, \_Traits, \_Allocator> &__y)`

#### 7.675.1 Detailed Description

This is a Standard C++ Library header.

## 7.676 stack File Reference

#### 7.676.1 Detailed Description

This is a Standard C++ Library header.

## 7.677 stdatomic.h File Reference

#### 7.677.1 Detailed Description

This is a Standard C++ Library header.

## 7.678 stdbit.h File Reference

#### 7.678.1 Detailed Description

This is a Standard C++ Library header.

## 7.679 stdckdint.h File Reference

### 7.679.1 Detailed Description

This is a Standard C++ Library header.

## 7.680 stdexcept File Reference

### Classes

- class [std::range\\_error](#)

### Namespaces

- namespace [std](#)

### 7.680.1 Detailed Description

This is a Standard C++ Library header.

## 7.681 stdlib.h File Reference

### Functions

- `template<typename _Tp>  
_Tp abs (const complex< _Tp > &)`
- `ldiv_t div (long __i, long __j) noexcept`

### 7.681.1 Detailed Description

This is a Standard C++ Library header.

### 7.681.2 Function Documentation

#### **abs()**

```
template<typename _Tp>
_Tp std::abs (
 const complex< _Tp > & __z) [inline]
Return magnitude of z.
```

## 7.682 stop\_token File Reference

### 7.682.1 Detailed Description

This is a Standard C++ Library header.

## 7.683 streambuf File Reference

### Classes

- class [std::basic\\_streambuf< \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)

## Macros

- `#define _IsUnused`

## Functions

- `template<typename _CharT, typename _Traits>`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< ↵`  
`_CharT, _Traits > *__sbout, bool &__ineof)`
- `template<> streamsize std::__copy_streambufs_eof (basic_streambuf< char > *__sbin, basic_streambuf<`  
`char > *__sbout, bool &__ineof)`
- `template<> streamsize std::__copy_streambufs_eof (basic_streambuf< wchar_t > *__sbin, basic_streambuf<`  
`wchar_t > *__sbout, bool &__ineof)`

### 7.683.1 Detailed Description

This is a Standard C++ Library header.

## 7.684 debug/string File Reference

### Classes

- class `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`
- struct `std::hash< __gnu_debug::basic_string< _CharT > >`

### Namespaces

- namespace `__gnu_debug`
- namespace `std`

### Macros

- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_constructor(_Str)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_string_n_constructor(_Str, _Size)`
- `#define _GLIBCXX_DEBUG_STRING`
- `#define _GLIBCXX_DEBUG_VERIFY_STR_COND_AT(_Cond, _File, _Line, _Func)`
- `#define _GLIBCXX_INSERT_RETURNS_ITERATOR`
- `#define _GLIBCXX_INSERT_RETURNS_ITERATOR_ONLY(expr)`

### Typedefs

- typedef `basic_string< char > __gnu_debug::string`
- typedef `basic_string< char16_t > __gnu_debug::u16string`
- typedef `basic_string< char32_t > __gnu_debug::u32string`
- typedef `basic_string< wchar_t > __gnu_debug::wstring`

### Functions

- `template<typename _CharT, typename _Integer>`  
`const _CharT * __gnu_debug::__check_string (const _CharT * __s, _Integer __n, const char * __file, unsigned int`  
`__line, const char * __function)`



- `template<typename _CharT>`  
`const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic\_istream< _CharT, _Traits > & \_\_gnu\_debug::getline (std::basic\_istream< _CharT, _Traits > & __is, basic\_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic\_istream< _CharT, _Traits > & \_\_gnu\_debug::getline (std::basic\_istream< _CharT, _Traits > & __is, basic\_string< _CharT, _Traits, _Allocator > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator!= (const _CharT * __lhs, const basic\_string< _CharT, _Traits, _Allocator > & __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator!= (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator!= (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const basic\_string< ↵  
_Chart, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic\_string< _CharT, _Traits, _Allocator > \_\_gnu\_debug::operator+ (_CharT __lhs, const basic\_string< _↵  
_Chart, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic\_string< _CharT, _Traits, _Allocator > \_\_gnu\_debug::operator+ (const _CharT * __lhs, const ↵  
basic\_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic\_string< _CharT, _Traits, _Allocator > \_\_gnu\_debug::operator+ (const basic\_string< _CharT, _Traits, ↵  
_Allocator > & __lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic\_string< _CharT, _Traits, _Allocator > \_\_gnu\_debug::operator+ (const basic\_string< _CharT, _Traits, ↵  
_Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`basic\_string< _CharT, _Traits, _Allocator > \_\_gnu\_debug::operator+ (const basic\_string< _CharT, _Traits, ↵  
_Allocator > & __lhs, const basic\_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator< (const _CharT * __lhs, const basic\_string< _CharT, _Traits, _Allocator > & __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator< (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator< (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const basic\_string< ↵  
_Chart, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic\_ostream< _CharT, _Traits > & \_\_gnu\_debug::operator<< (std::basic\_ostream< _CharT, _Traits > ↵  
& __os, const basic\_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator<= (const _CharT * __lhs, const basic\_string< _CharT, _Traits, _Allocator > & __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator<= (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool \_\_gnu\_debug::operator<= (const basic\_string< _CharT, _Traits, _Allocator > & __lhs, const basic\_string< ↵  
_Chart, _Traits, _Allocator > & __rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`bool __gnu_debug::operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`std::basic\_istream< _CharT, _Traits > & __gnu_debug::operator>> (std::basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>`  
`void __gnu_debug::swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

### 7.684.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.685 experimental/string File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define \_GLIBCXX\_EXPERIMENTAL\_STRING`

### Typedefs

- `template<typename _CharT, typename _Traits = char\_traits<_CharT>>`  
`using std::experimental::fundamentals\_v2::pmr::basic\_string`
- `typedef basic\_string< char > std::experimental::fundamentals\_v2::pmr::string`

- typedef basic\_string< char16\_t > **std::experimental::fundamentals\_v2::pmr::u16string**
- typedef basic\_string< char32\_t > **std::experimental::fundamentals\_v2::pmr::u32string**
- typedef basic\_string< wchar\_t > **std::experimental::fundamentals\_v2::pmr::wstring**

## Functions

- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_Up>  
void **std::experimental::erase** (basic\_string< \_CharT, \_Traits, \_Alloc > &\_\_cont, const \_Up &\_\_value)
- template<typename \_CharT, typename \_Traits, typename \_Alloc, typename \_Predicate>  
void **std::experimental::erase\_if** (basic\_string< \_CharT, \_Traits, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 7.685.1 Detailed Description

This is a TS C++ Library header.

## 7.686 string File Reference

### Namespaces

- namespace [std](#)

### Macros

- #define **\_\_glibcxx\_exc\_in\_string**
- #define **\_GLIBCXX\_STRING**

### Typedefs

- template<typename \_CharT, typename \_Traits = char\_traits<\_CharT>>  
using **std::pmr::basic\_string**
- using **std::pmr::string**
- using **std::pmr::u16string**
- using **std::pmr::u32string**
- using **std::pmr::wstring**

### 7.686.1 Detailed Description

This is a Standard C++ Library header.

## 7.687 experimental/string\_view File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view< \\_CharT, \\_Traits >](#)

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- #define **\_\_cpp\_lib\_experimental\_string\_view**
- #define **\_GLIBCXX\_EXPERIMENTAL\_STRING\_VIEW**

## Typedefs

- using **std::experimental::string\_view**
- using **std::experimental::u16string\_view**
- using **std::experimental::u32string\_view**
- using **std::experimental::wstring\_view**

## Functions

- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator!=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- constexpr [basic\\_string\\_view](#)< char > **std::experimental::literals::operator""sv** (const char \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char16\_t > **std::experimental::literals::operator""sv** (const char16\_t \*\_\_str,  
size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< char32\_t > **std::experimental::literals::operator""sv** (const char32\_t \*\_\_str,  
size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)< wchar\_t > **std::experimental::literals::operator""sv** (const wchar\_t \*\_\_str,  
size\_t \_\_len) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<** ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)< \_CharT, \_Traits > & **std::experimental::operator<<** ([basic\\_ostream](#)< \_CharT, \_Traits > &<←  
\_\_os, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_str)
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<=** ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator<=** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [basic\\_string\\_view](#)<  
\_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator==** ([\\_\\_type\\_identity\\_t](#)< [basic\\_string\\_view](#)< \_CharT, \_Traits > > <←  
\_\_x, [basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_y) noexcept
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::experimental::operator==** ([basic\\_string\\_view](#)< \_CharT, \_Traits > \_\_x, [\\_\\_type\\_identity\\_t](#)<  
[basic\\_string\\_view](#)< \_CharT, \_Traits > > \_\_y) noexcept

- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator== (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator> (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= ( __type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::experimental::operator>= (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept`

## Variables

- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::ranges::enable_borrowed_range< experimental::basic_string_view< _CharT, _Traits > >`
- `template<typename _CharT, typename _Traits>`  
`constexpr bool std::ranges::enable_view< experimental::basic_string_view< _CharT, _Traits > >`

## 7.687.1 Detailed Description

This is a TS C++ Library header.

## 7.688 string\_view File Reference

### Classes

- class `std::basic_string_view< _CharT, _Traits >`

### Namespaces

- namespace `std`
- namespace `std::literals`

### Macros

- `#define __glibcxx_exc_in_string`

### Typedefs

- using `std::string_view`
- using `std::u16string_view`
- using `std::u32string_view`
- using `std::wstring_view`

## Functions

- constexpr size\_t **std::\_\_sv\_check** (size\_t \_\_size, size\_t \_\_pos, const char \*\_\_s)
- constexpr size\_t **std::\_\_sv\_limit** (size\_t \_\_size, size\_t \_\_pos, size\_t \_\_off) noexcept
- template<contiguous\_iterator \_It, sized\_sentinel\_for<\_It> \_End>  
**std::basic\_string\_view** (\_It, \_End) -> basic\_string\_view<iter\_value\_t<\_It>>
- template<ranges::contiguous\_range \_Range>  
**std::basic\_string\_view** (\_Range &&) -> basic\_string\_view<ranges::range\_value\_t<\_Range>>
- constexpr [basic\\_string\\_view](#)<char> **std::literals::operator""sv** (const char \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)<char16\_t> **std::literals::operator""sv** (const char16\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)<char32\_t> **std::literals::operator""sv** (const char32\_t \*\_\_str, size\_t \_\_len) noexcept
- constexpr [basic\\_string\\_view](#)<wchar\_t> **std::literals::operator""sv** (const wchar\_t \*\_\_str, size\_t \_\_len) noexcept
- template<typename \_CharT, typename \_Traits>  
[basic\\_ostream](#)<\_CharT, \_Traits> & **std::operator**<< ([basic\\_ostream](#)<\_CharT, \_Traits> &\_\_os, [basic\\_string\\_view](#)<\_CharT, \_Traits> \_\_str)
- template<typename \_CharT, typename \_Traits>  
constexpr auto **std::operator**<=> ([basic\\_string\\_view](#)<\_CharT, \_Traits> \_\_x, type\_identity\_t<[basic\\_string\\_view](#)<\_CharT, \_Traits>> \_\_y) noexcept -> decltype(\_\_detail::\_\_char\_traits\_cmp\_cat<\_Traits>(0))
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::operator**== ([basic\\_string\\_view](#)<\_CharT, \_Traits> \_\_x, type\_identity\_t<[basic\\_string\\_view](#)<\_CharT, \_Traits>> \_\_y) noexcept

## Variables

- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::ranges::enable\_borrowed\_range**<[basic\\_string\\_view](#)<\_CharT, \_Traits>>
- template<typename \_CharT, typename \_Traits>  
constexpr bool **std::ranges::enable\_view**<[basic\\_string\\_view](#)<\_CharT, \_Traits>>

### 7.688.1 Detailed Description

This is a Standard C++ Library header.

## 7.689 syncstream File Reference

### 7.689.1 Detailed Description

This is a Standard C++ Library header.

## 7.690 experimental/system\_error File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- #define [\\_GLIBCXX\\_EXPERIMENTAL\\_SYSTEM\\_ERROR](#)

## Variables

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_error_code_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_error_condition_enum_v`

### 7.690.1 Detailed Description

This is a TS C++ Library header.

## 7.691 `system_error` File Reference

### Classes

- class `std::error_category`
- class `std::error_code`
- class `std::error_condition`
- struct `std::hash< error_code >`
- struct `std::hash< error_condition >`
- struct `std::is_error_code_enum< _Tp >`
- struct `std::is_error_condition_enum< _Tp >`
- class `std::system_error`

### Namespaces

- namespace `std`

### Functions

- `const error_category & std::generic_category () noexcept`
- `void std::__adl_only::make_error_code ()=delete`
- `void std::__adl_only::make_error_condition ()=delete`
- `const error_category & std::system_category () noexcept`

## Variables

- `template<typename _Tp>`  
`constexpr bool std::is_error_code_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::is_error_condition_enum_v`

### 7.691.1 Detailed Description

This is a Standard C++ Library header.

## 7.692 `text_encoding` File Reference

### 7.692.1 Detailed Description

This is a Standard C++ Library header.

## 7.693 `tgmath.h` File Reference

### 7.693.1 Detailed Description

This is a Standard C++ Library header.

## 7.694 thread File Reference

### Namespaces

- namespace [std](#)

### 7.694.1 Detailed Description

This is a Standard C++ Library header.

## 7.695 bool\_set File Reference

### Classes

- class [std::tr2::bool\\_set](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET`

### Functions

- `bool std::tr2::certainly (bool_set __b)`
- `bool std::tr2::contains (bool_set __s, bool_set __t)`
- `bool std::tr2::equals (bool_set __s, bool_set __t)`
- `bool std::tr2::is_emptyset (bool_set __b)`
- `bool std::tr2::is_indeterminate (bool_set __b)`
- `bool std::tr2::is_singleton (bool_set __b)`
- `bool_set std::tr2::operator!= (bool __s, bool_set __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool __t)`
- `bool_set std::tr2::operator!= (bool_set __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool __s, bool_set __t)`
- `bool_set std::tr2::operator& (bool_set __s, bool __t)`
- `bool_set std::tr2::operator== (bool __s, bool_set __t)`
- `bool_set std::tr2::operator== (bool_set __s, bool __t)`
- `bool_set std::tr2::operator^ (bool __s, bool_set __t)`
- `bool_set std::tr2::operator^ (bool_set __s, bool __t)`
- `bool_set std::tr2::operator| (bool __s, bool_set __t)`
- `bool_set std::tr2::operator| (bool_set __s, bool __t)`
- `bool std::tr2::possibly (bool_set __b)`
- `bool_set std::tr2::set_complement (bool_set __b)`
- `bool_set std::tr2::set_intersection (bool __s, bool_set __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool __t)`
- `bool_set std::tr2::set_intersection (bool_set __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool __s, bool_set __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool __t)`
- `bool_set std::tr2::set_union (bool_set __s, bool_set __t)`



### 7.695.1 Detailed Description

This is a TR2 C++ Library header.

## 7.696 bool\_set.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### 7.696.1 Detailed Description

This is a TR2 C++ Library header.

## 7.697 dynamic\_bitset File Reference

### Classes

- struct [std::tr2::\\_\\_dynamic\\_bitset\\_base< \\_WordT, \\_Alloc >](#)
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >](#)
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >::reference](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define \_GLIBCXX\_TR2\_DYNAMIC\_BITSET`

### Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>  
std::basic\_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic\_ostream< _CharT, _Traits > &__os,  
const dynamic\_bitset< _WordT, _Alloc > &__x)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator!= (const dynamic\_bitset< _WordT, _Alloc > &__lhs, const dynamic\_bitset< _WordT, _Alloc  
> &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator<= (const dynamic\_bitset< _WordT, _Alloc > &__lhs, const dynamic\_bitset< _WordT, _Alloc  
> &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator> (const dynamic\_bitset< _WordT, _Alloc > &__lhs, const dynamic\_bitset< _WordT, _Alloc  
> &__rhs)`
- `template<typename _WordT, typename _Alloc>  
bool std::tr2::operator>= (const dynamic\_bitset< _WordT, _Alloc > &__lhs, const dynamic\_bitset< _WordT, _Alloc  
> &__rhs)`
- `template<typename _WordT, typename _Alloc>  
dynamic\_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic\_bitset< _WordT, _Alloc > &__x, const  
dynamic\_bitset< _WordT, _Alloc > &__y)`

- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator-` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)
- `template<typename _WordT, typename _Alloc>`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator|` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)

### 7.697.1 Detailed Description

This is a TR2 C++ Library header.

## 7.698 dynamic\_bitset.tcc File Reference

### Namespaces

- namespace `std`
- namespace `std::tr2`

### Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>`  
`std::basic_istream< _CharT, _Traits > & std::tr2::operator>>` (`std::basic_istream< _CharT, _Traits > &__is`, `dynamic_bitset< _WordT, _Alloc > &__x`)

### 7.698.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<tr2/dynamic_bitset>`.

## 7.699 experimental/tuple File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define __cpp_lib_experimental_tuple`
- `#define _GLIBCXX_EXPERIMENTAL_TUPLE`

### Functions

- `template<typename _Fn, typename _Tuple, std::size_t... _Idx>`  
`constexpr decltype(auto) std::experimental::__apply_impl` (`_Fn &&__f`, `_Tuple &&__t`, `std::index_sequence< _Idx... >`)
- `template<typename _Fn, typename _Tuple>`  
`constexpr decltype(auto) std::experimental::apply` (`_Fn &&__f`, `_Tuple &&__t`)

### Variables

- `template<typename _Tp>`  
`constexpr size_t std::experimental::tuple_size_v`

### 7.699.1 Detailed Description

This is a TS C++ Library header.

## 7.700 tuple File Reference

### Classes

- class [std::tuple< \\_Elements >](#)
- class [std::tuple< \\_T1, \\_T2 >](#)
- struct [std::tuple\\_element< \\_\\_i, tuple< \\_Types... > >](#)
- struct [std::tuple\\_size< tuple< \\_Elements... > >](#)
- struct [std::uses\\_allocator< tuple< \\_Types... >, \\_Alloc >](#)

### Namespaces

- namespace [std](#)

### Functions

- [template<size\\_t \\_\\_i, typename \\_Head, typename... \\_Tail>  
constexpr \\_Head & std::\\_\\_get\\_helper \( \\_Tuple\\_impl< \\_\\_i, \\_Head, \\_Tail... > &\\_\\_t\) noexcept](#)
- [template<size\\_t \\_\\_i, typename \\_Head, typename... \\_Tail>  
constexpr const \\_Head & std::\\_\\_get\\_helper \(const \\_Tuple\\_impl< \\_\\_i, \\_Head, \\_Tail... > &\\_\\_t\) noexcept](#)
- [template<size\\_t \\_\\_i, typename... \\_Types>  
\\_\\_enable\\_if\\_t<\(\\_\\_i >= sizeof...\( \\_Types\)\)> std::\\_\\_get\\_helper \(const \[tuple< \\\_Types... >\]\(#\) &\)=delete](#)
- [template<typename \\_Cat, typename \\_Tp, typename \\_Up, typename \\_IndexSeq>  
constexpr \\_Cat std::tuple\\_cmp \(const \\_Tp &\\_\\_t, const \\_Up &\\_\\_u, \\_IndexSeq \\_\\_indices\)](#)
- [template<typename... \\_Elements>  
constexpr \[tuple< \\\_Elements &&... >\]\(#\) std::forward\\_as\\_tuple \( \\_Elements &&... \\_\\_args\) noexcept](#)
- [template<size\\_t \\_\\_i, typename... \\_Elements>  
constexpr const \\_\\_tuple\\_element\\_t< \\_\\_i, \[tuple< \\\_Elements... >\]\(#\) > && std::get \(const \[tuple< \\\_Elements... >\]\(#\) &&\\_\\_t\) noexcept](#)
- [template<size\\_t \\_\\_i, typename... \\_Elements>  
constexpr const \\_\\_tuple\\_element\\_t< \\_\\_i, \[tuple< \\\_Elements... >\]\(#\) > & std::get \(const \[tuple< \\\_Elements... >\]\(#\) &\\_\\_t\) noexcept](#)
- [template<size\\_t \\_\\_i, typename... \\_Elements>  
constexpr \\_\\_tuple\\_element\\_t< \\_\\_i, \[tuple< \\\_Elements... >\]\(#\) > && std::get \( \[tuple< \\\_Elements... >\]\(#\) &&\\_\\_t\) noexcept](#)
- [template<size\\_t \\_\\_i, typename... \\_Elements>  
constexpr \\_\\_tuple\\_element\\_t< \\_\\_i, \[tuple< \\\_Elements... >\]\(#\) > & std::get \( \[tuple< \\\_Elements... >\]\(#\) &\\_\\_t\) noexcept](#)
- [template<typename... \\_Elements>  
constexpr \[tuple< typename \\\_\\\_decay\\\_and\\\_strip< \\\_Elements >::\\\_\\\_type... >\]\(#\) std::make\\_tuple \( \\_Elements &&... \\_\\_args\)](#)
- [template<typename... \\_Tps, typename... \\_Ups>  
constexpr common\\_comparison\\_category\\_t< \\_\\_detail::\\_\\_synth3way\\_t< \\_Tps, \\_Ups >... > std::operator<=> \(const \[tuple< \\\_Tps... >\]\(#\) &\\_\\_t, const \[tuple< \\\_Ups... >\]\(#\) &\\_\\_u\)](#)
- [template<typename... \\_Tps, typename... \\_Ups>  
constexpr bool std::operator== \(const \[tuple< \\\_Tps... >\]\(#\) &\\_\\_t, const \[tuple< \\\_Ups... >\]\(#\) &\\_\\_u\)](#)
- [template<typename... \\_Elements>  
constexpr \[enable\\\_if<! \\\_\\\_and\\\_< \\\_\\\_is\\\_swappable< \\\_Elements >... >::value >::type\]\(#\) std::swap \( \[tuple< \\\_Elements... >\]\(#\) &\\_\\_, \[tuple< \\\_Elements... >\]\(#\) &\\_\\_\)=delete](#)
- [template<typename... \\_Elements>  
constexpr \[enable\\\_if< \\\_\\\_and\\\_< \\\_\\\_is\\\_swappable< \\\_Elements >... >::value >::type\]\(#\) std::swap \( \[tuple< \\\_Elements... >\]\(#\) &\\_\\_x, \[tuple< \\\_Elements... >\]\(#\) &\\_\\_y\) noexcept\(noexcept\(\\_\\_x.swap\(\\_\\_y\)\)\)](#)

- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > std::tie ( _Elements &... __args) noexcept`
- `template<typename... _UTypes>`  
`std::tuple ( _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >`
- `template<typename _Alloc, typename _T1, typename _T2>`  
`std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename _Alloc, typename... _UTypes>`  
`std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >`
- `template<typename _T1, typename _T2>`  
`std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto std::tuple_cat ( _Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type`

## Variables

- `template<typename... _Types>`  
`constexpr size_t std::tuple_size_v< const tuple< _Types... > >`
- `template<typename... _Types>`  
`constexpr size_t std::tuple_size_v< tuple< _Types... > >`

### 7.700.1 Detailed Description

This is a Standard C++ Library header.

## 7.701 experimental/type\_traits File Reference

### Namespaces

- namespace `std`
- namespace `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_TYPE_TRAITS`
- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::alignment_of_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::extent_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::has_virtual_destructor_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_abstract_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_arithmetic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_array_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_assignable_v`
- `template<typename _Base, typename _Derived>`  
`constexpr bool std::experimental::is_base_of_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_class_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_compound_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_const_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_constructible_v`
- `template<typename _From, typename _To>`  
`constexpr bool std::experimental::is_convertible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_empty_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_enum_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_final_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_floating_point_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_function_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_fundamental_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_integral_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_literal_type_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_lvalue_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_function_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_object_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_member_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_move_constructible_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_nothrow_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_nothrow_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_assignable_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_nothrow_move_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_null_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_object_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pod_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_pointer_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_polymorphic_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_reference_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_rvalue_reference_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_same_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_same_v< _Tp, _Tp >`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_scalar_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_signed_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_standard_layout_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivial_v`
- `template<typename _Tp, typename _Up>`  
`constexpr bool std::experimental::is_trivially_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::is_trivially_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_assignable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copy_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_copyable_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_default_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_destructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_assignable_v`

- `template<typename _Tp>`  
`constexpr bool std::experimental::is_trivially_move_constructible_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_union_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_unsigned_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_void_v`
- `template<typename _Tp>`  
`constexpr bool std::experimental::is_volatile_v`
- `template<typename _Tp>`  
`constexpr size_t std::experimental::rank_v`
- `#define __cpp_lib_experimental_detect`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected_or`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected_or_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::detected_t`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is_detected`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is_detected_convertible`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is_detected_convertible_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::is_detected_exact`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is_detected_exact_v`
- `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::is_detected_v`
- `template<typename...>`  
`using std::experimental::void_t`
- `#define __cpp_lib_experimental_logical_traits`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::conjunction_v`
- `template<typename... _Bn>`  
`constexpr bool std::experimental::disjunction_v`
- `template<typename _Pp>`  
`constexpr bool std::experimental::negation_v`

### 7.701.1 Detailed Description

This is a TS C++ Library header.

This header defines variable templates for the C++14 type traits.

Equivalent variable templates are defined in namespace `std` since C++17.

See also

`variable_templates`

Since

C++14

## 7.702 tr2/type\_traits File Reference

### Classes

- struct [std::tr2::\\_\\_reflection\\_typelist<\\_First, \\_Rest... >](#)
- struct [std::tr2::\\_\\_reflection\\_typelist<>](#)
- struct [std::tr2::bases<\\_Tp >](#)
- struct [std::tr2::direct\\_bases<\\_Tp >](#)

### Namespaces

- namespace [std](#)
- namespace [std::tr2](#)

### Macros

- `#define \_GLIBCXX\_TR2\_TYPE\_TRAITS`

#### 7.702.1 Detailed Description

This is a TR2 C++ Library header.

## 7.703 type\_traits File Reference

### Classes

- struct [std::add\\_const<\\_Tp >](#)
- struct [std::add\\_cv<\\_Tp >](#)
- struct [std::add\\_lvalue\\_reference<\\_Tp >](#)
- struct [std::add\\_pointer<\\_Tp >](#)
- struct [std::add\\_rvalue\\_reference<\\_Tp >](#)
- struct [std::add\\_volatile<\\_Tp >](#)
- struct [std::aligned\\_storage<\\_Len, \\_Align >](#)
- struct [std::aligned\\_union<\\_Len, \\_Types >](#)
- struct [std::alignment\\_of<\\_Tp >](#)
- struct [std::conditional<\\_Cond, \\_Iftrue, \\_Iffalse >](#)
- struct [std::enable\\_if<bool, \\_Tp >](#)
- struct [std::extent<typename, \\_UInt >](#)
- struct [std::has\\_virtual\\_destructor<\\_Tp >](#)
- struct [std::integral\\_constant<\\_Tp, \\_\\_v >](#)
- struct [std::is\\_abstract<\\_Tp >](#)
- struct [std::is\\_arithmetic<\\_Tp >](#)
- struct [std::is\\_array<\\_Tp >](#)
- struct [std::is\\_assignable<\\_Tp, \\_Up >](#)
- struct [std::is\\_base\\_of<\\_Base, \\_Derived >](#)
- struct [std::is\\_class<\\_Tp >](#)
- struct [std::is\\_compound<\\_Tp >](#)
- struct [std::is\\_const<\\_Tp >](#)
- struct [std::is\\_constructible<\\_Tp, \\_Args >](#)
- struct [std::is\\_copy\\_assignable<\\_Tp >](#)
- struct [std::is\\_copy\\_constructible<\\_Tp >](#)
- struct [std::is\\_default\\_constructible<\\_Tp >](#)
- struct [std::is\\_destructible<\\_Tp >](#)
- struct [std::is\\_empty<\\_Tp >](#)



- struct `std::is_enum< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_layout_compatible< _Tp, _Up >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_pointer_interconvertible_base_of< _Base, _Derived >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_signed< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_copyable< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_unsigned< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< _Tp >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`

- struct [std::remove\\_cv<\\_Tp>](#)
- struct [std::remove\\_extent<\\_Tp>](#)
- struct [std::remove\\_pointer<\\_Tp>](#)
- struct [std::remove\\_reference<\\_Tp>](#)
- struct [std::remove\\_volatile<\\_Tp>](#)
- struct [std::underlying\\_type<\\_Tp>](#)

## Namespaces

- namespace [std](#)

## Typedefs

- template<bool \_Cond, typename \_If, typename \_Else>  
using **std::\_\_conditional\_t**
- template<bool \_Cond, typename \_Tp = void>  
using **std::\_\_enable\_if\_t**
- template<typename \_ToElementType, typename \_FromElementType>  
using **std::\_\_is\_array\_convertible**
- template<typename \_Tp>  
using [std::add\\_lvalue\\_reference\\_t](#)
- template<typename \_Tp>  
using [std::add\\_pointer\\_t](#)
- template<typename \_Tp>  
using [std::add\\_rvalue\\_reference\\_t](#)
- template<size\_t \_Len, size\_t \_Align = \_\_aligned\_storage\_default\_alignment(\_Len)>  
using [std::aligned\\_storage\\_t](#)
- template<size\_t \_Len, typename... \_Types>  
using **std::aligned\_union\_t**
- template<typename... \_Tp>  
using [std::common\\_type\\_t](#)
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse>  
using [std::conditional\\_t](#)
- template<typename \_Tp>  
using [std::decay\\_t](#)
- template<bool \_Cond, typename \_Tp = void>  
using [std::enable\\_if\\_t](#)
- using [std::false\\_type](#)
- template<typename \_Tp>  
using [std::make\\_signed\\_t](#)
- template<typename \_Tp>  
using [std::make\\_unsigned\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_all\\_extents\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_extent\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_pointer\\_t](#)
- template<typename \_Tp>  
using [std::remove\\_reference\\_t](#)
- template<typename \_Tp>  
using [std::result\\_of\\_t](#)
- using [std::true\\_type](#)

- `template<typename _Tp>`  
using [std::underlying\\_type\\_t](#)

## Functions

- `template<typename _Tp>`  
auto [std::declval](#) () noexcept -> decltype(\_\_declval< \_Tp >())
- `template<typename _Tp>`  
constexpr `_Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > >` [std::swap](#) (\_Tp &, \_Tp &) noexcept(\_\_and\_< [is\\_nothrow\\_move\\_constructible](#)< \_Tp >, [is\\_nothrow\\_move\\_assignable](#)< \_Tp > >::value)
- `template<typename _Tp, size_t _Nm>`  
constexpr `__enable_if_t< __is_swappable< _Tp >::value >` [std::swap](#) (\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm])↔  
) noexcept(\_\_is\_nothrow\_swappable< \_Tp >::value)

## Variables

- `template<typename _Tp, typename... _Args>`  
constexpr bool **`std::__is_nothrow_new_constructible`**
- `template<size_t _Len, typename... _Types>`  
const size\_t **`std::aligned_union< _Len, _Types... >::alignment_value`**
- `template<typename... _Tp>`  
using [std::common\\_reference\\_t](#)
- `template<typename _S1, typename _S2, typename _M1, typename _M2>`  
constexpr bool [std::is\\_corresponding\\_member](#) (\_M1 \_S1::\*\_\_m1, \_M2 \_S2::\*\_\_m2) noexcept
- `template<typename _Tp, typename _Up>`  
constexpr bool [std::is\\_layout\\_compatible\\_v](#)
- `template<typename _Base, typename _Derived>`  
constexpr bool [std::is\\_pointer\\_interconvertible\\_base\\_of\\_v](#)
- `template<typename _Tp, typename _Mem>`  
constexpr bool [std::is\\_pointer\\_interconvertible\\_with\\_class](#) (\_Mem \_Tp::\*\_\_mp) noexcept

### 7.703.1 Detailed Description

This is a Standard C++ Library header.

## 7.704 typeindex File Reference

### Classes

- struct [std::hash< type\\_index >](#)
- struct [std::type\\_index](#)

### Namespaces

- namespace [std](#)

### 7.704.1 Detailed Description

This is a Standard C++ Library header.

## 7.705 debug/unordered\_map File Reference

### Classes

- class `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

### Namespaces

- namespace `std`
- namespace `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map<int, int>::size_type, _Allocator) -> unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, hash<__iter_key_t<_InputIterator>>, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map<int, int>::size_type, _Hash, _Allocator) -> unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, equal_to<__iter_key_t<_InputIterator>>, _Allocator>`
- `template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map<int, int>::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map<__iter_key_t<_InputIterator>, __iter_val_t<_InputIterator>, _Hash, _Pred, _Allocator>`
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>>`  
`std::__debug::unordered_map (initializer_list<pair<_Key, _Tp>>, _Allocator) -> unordered_map<_Key, _Tp, hash<_Key>, equal_to<_Key>, _Allocator>`

- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `typename unordered_map`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_map`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `typename unordered_map`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_map`< `_Key`, `_Tp`, `_Hash`, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_map** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `typename unordered_map`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_map`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `_Allocator`) -> `unordered_multimap`< `_iter_key_t`< `_InputIterator` >, `_iter_val_t`< `_InputIterator` >, `hash`< `_iter_key_t`< `_InputIterator` > >, `equal_to`< `_iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_multimap`< `_iter_key_t`< `_InputIterator` >, `_iter_val_t`< `_InputIterator` >, `hash`< `_iter_key_t`< `_InputIterator` > >, `equal_to`< `_iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap`< `_iter_key_t`< `_InputIterator` >, `_iter_val_t`< `_InputIterator` >, `_Hash`, `equal_to`< `_iter_key_t`< `_InputIterator` > >, `_Allocator` >
- `template<typename _InputIterator, typename _Hash = hash<_iter_key_t<_InputIterator>>, typename _Pred = equal_to<_iter_key_t<_InputIterator>>, typename _Allocator = allocator<_iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`_InputIterator`, `_InputIterator`, `unordered_multimap`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multimap`< `_iter_key_t`< `_InputIterator` >, `_iter_val_t`< `_InputIterator` >, `_Hash`, `_Pred`, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `hash`< `_Key` >, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`, `_Hash`, `_Allocator`) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `equal_to`< `_Key` >, `_Allocator` >
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>`  
**std::debug::unordered\_multimap** (`initializer_list`< `pair`< `_Key`, `_Tp` > >, `unordered_multimap`< `int`, `int` >::`size_type`={}, `_Hash`=`_Hash`(), `_Pred`=`_Pred`(), `_Allocator`=`_Allocator`()) -> `unordered_multimap`< `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Allocator` >

### 7.705.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 7.706 experimental/unordered\_map File Reference

### Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_MAP`

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
`using std::experimental::fundamentals_v2::pmr::unordered_map`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
`using std::experimental::fundamentals_v2::pmr::unordered_multimap`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate`  
`__pred)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`  
`void std::experimental::erase_if (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _↵`  
`Predicate __pred)`

### 7.706.1 Detailed Description

This is a TS C++ Library header.

## 7.707 unordered\_map File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>`  
`using std::pmr::unordered_map`
- `template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>`  
`using std::pmr::unordered_multimap`

### 7.707.1 Detailed Description

This is a Standard C++ Library header.

## 7.708 debug/unordered\_set File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)
- class [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Macros

- `#define \_GLIBCXX\_DEBUG\_UNORDERED\_SET`

### Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>  
bool std::\_\_debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>  
bool std::\_\_debug::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>  
void std::\_\_debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>  
void std::\_\_debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
std::\_\_debug::unordered\_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator\_traits< _InputIterator >::value_type, hash< typename iterator\_traits< _InputIterator >::value_type >, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>  
std::\_\_debug::unordered\_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator\_traits< _InputIterator >::value_type, hash< typename iterator\_traits< _InputIterator >::value_type >, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>  
std::\_\_debug::unordered\_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator\_traits< _InputIterator >::value_type, _Hash, equal\_to< typename iterator\_traits< _InputIterator >::value_type >, _Allocator >`
- `template<typename _InputIterator, typename _Hash = hash<typename iterator\_traits<_InputIterator>::value_type>, typename _Pred = equal\_to<typename iterator\_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator\_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocatorOrPred>, typename = _RequireAllocator<_Allocator>>  
std::\_\_debug::unordered\_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator\_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >`

- ```

template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash<
_Tp >, equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator)
-> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _↵
RequireAllocator<_Allocator>>
std:: debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _↵
Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↵
Allocator>>
std:: debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_↵
_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵
Allocator<_Allocator>>
std:: debug::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename
iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type
>, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↵
Allocator<_Allocator>>
std:: debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) ->
unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _↵
InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator
>

• template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _↵
RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _↵
Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< type-
name iterator_traits< _InputIterator >::value_type >, _Allocator >

• template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred
= equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↵
Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename
= _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_↵
Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator
>::value_type, _Hash, _Pred, _Allocator >

• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >,
equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
std:: debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) ->
unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _↵
RequireAllocator<_Allocator>>
std:: debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) ->
unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >

• template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>,
typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↵
Allocator>>
std:: debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _↵
Pred=_Pred(), _Allocator=_Allocator()) -> unordered set< Tp. Hash. Pred. Allocator >

```


7.708.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.709 experimental/unordered_set File Reference

Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_SET`

Typedefs

- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`
`using std::experimental::fundamentals_v2::pmr::unordered_multiset`
- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>>`
`using std::experimental::fundamentals_v2::pmr::unordered_set`

Functions

- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`
`void std::experimental::erase_if (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>`
`void std::experimental::erase_if (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)`

7.709.1 Detailed Description

This is a TS C++ Library header.

7.710 unordered_set File Reference

Namespaces

- namespace [std](#)

Typedefs

- `template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>>`
`using std::pmr::unordered_multiset`
- `template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>>`
`using std::pmr::unordered_set`

7.710.1 Detailed Description

This is a Standard C++ Library header.

7.711 experimental/utility File Reference

Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

Macros

- `#define _GLIBCXX_EXPERIMENTAL_UTILITY`

Typedefs

- using `std::experimental::erased_type`

7.711.1 Detailed Description

This is a TS C++ Library header.

7.712 utility File Reference

Namespaces

- namespace [std](#)

7.712.1 Detailed Description

This is a Standard C++ Library header.

7.713 valarray File Reference

Classes

- class [std::valarray<_Tp>](#)

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Functions

- `template<typename _Tp, size_t _Nm>
std::valarray (const _Tp(&)[_Nm], size_t) -> valarray<_Tp>`

7.713.1 Detailed Description

This is a Standard C++ Library header.

7.714 variant File Reference

7.714.1 Detailed Description

This is the `<variant>` C++ Library header.

7.715 debug/vector File Reference

Classes

- class [__gnu_debug::__Safe_vector<_SafeSequence, _BaseSequence>](#)
- struct [std::hash<__debug::vector<bool, _Alloc>>](#)
- class [std::__debug::vector<_Tp, _Allocator>](#)

Namespaces

- namespace [__gnu_debug](#)
- namespace [std](#)
- namespace [std::__debug](#)
- namespace [std::__detail](#)

Macros

- `#define _GLIBCXX_DEBUG_VECTOR`

Functions

- `template<typename _Tp, typename _Alloc>
constexpr __detail::__synth3way_t< _Tp > std::__debug::operator<=> (const vector< _Tp, _Alloc > &__x,
const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc>
constexpr bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc >
&__rhs)`
- `template<typename _Tp, typename _Alloc>
constexpr void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(↩
*conditional *)`
- `template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator =
allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
std::__debug::vector (_InputIterator, _InputIterator, _Allocator= _Allocator()) -> vector< _ValT, _Allocator >`
- `template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
std::__debug::vector (size_t, _Tp, _Allocator= _Allocator()) -> vector< _Tp, _Allocator >`

7.715.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.716 experimental/vector File Reference

Namespaces

- namespace [std](#)
- namespace [std::experimental](#)

Macros

- `#define __cpp_lib_experimental_erase_if`
- `#define _GLIBCXX_EXPERIMENTAL_VECTOR`

Typedefs

- `template<typename _Tp>
using std::experimental::fundamentals_v2::pmr::vector`

Functions

- `template<typename _Tp, typename _Alloc, typename _Up>
void std::experimental::erase (vector< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate>
void std::experimental::erase_if (vector< _Tp, _Alloc > &__cont, _Predicate __pred)`

7.716.1 Detailed Description

This is a TS C++ Library header.

7.717 vector File Reference

Namespaces

- namespace [std](#)

Typedefs

- `template<typename _Tp>`
using **`std::pmr::vector`**

7.717.1 Detailed Description

This is a Standard C++ Library header.

7.718 atomic_word.h File Reference

Macros

- `#define _GLIBCXX_READ_MEM_BARRIER`
- `#define _GLIBCXX_WRITE_MEM_BARRIER`

Typedefs

- `typedef int _Atomic_word`

7.718.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.719 basic_file.h File Reference

Namespaces

- namespace [std](#)

7.719.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

7.720 c++allocator.h File Reference

Namespaces

- namespace [std](#)

Typedefs

- `template<typename _Tp>`
using [std::__allocator_base](#)

7.720.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

7.721 c++config.h File Reference

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(cond)`
- `#define __N(msgid)`
- `#define _GLIBCXX11_DEPRECATED`
- `#define _GLIBCXX11_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX11_USE_C99_MATH`
- `#define _GLIBCXX11_USE_C99_STDIO`
- `#define _GLIBCXX11_USE_C99_STDLIB`
- `#define _GLIBCXX11_USE_C99_WCHAR`
- `#define _GLIBCXX14_DEPRECATED`
- `#define _GLIBCXX14_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX17_DEPRECATED`
- `#define _GLIBCXX17_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX20_DEPRECATED`
- `#define _GLIBCXX20_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX23_DEPRECATED`
- `#define _GLIBCXX23_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX26_CONSTEXPR`
- `#define _GLIBCXX26_DEPRECATED`
- `#define _GLIBCXX26_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX98_USE_C99_COMPLEX`
- `#define _GLIBCXX98_USE_C99_MATH`
- `#define _GLIBCXX98_USE_C99_STDIO`
- `#define _GLIBCXX98_USE_C99_STDLIB`
- `#define _GLIBCXX98_USE_C99_WCHAR`
- `#define _GLIBCXX_ASSERT_FAIL(_Condition)`
- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_ATOMIC_WORD_BUILTINS`
- `#define _GLIBCXX_AUTO_CAST(X)`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE(X)`
- `#define _GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE`

- #define _GLIBCXX_DARWIN_USE_64_BIT_INODE
- #define _GLIBCXX_DEFAULT_ABI_TAG
- #define _GLIBCXX_DEPRECATED
- #define _GLIBCXX_DEPRECATED_SUGGEST(ALT)
- #define _GLIBCXX_DOXYGEN_ONLY(X)
- #define _GLIBCXX_END_EXTERN_C
- #define _GLIBCXX_END_INLINE_ABI_NAMESPACE(X)
- #define _GLIBCXX_END_NAMESPACE_ALGO
- #define _GLIBCXX_END_NAMESPACE_CONTAINER
- #define _GLIBCXX_END_NAMESPACE_CXX11
- #define _GLIBCXX_END_NAMESPACE_LDBL
- #define _GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11
- #define _GLIBCXX_END_NAMESPACE_VERSION
- #define _GLIBCXX_EXTERN_TEMPLATE
- #define _GLIBCXX_EXTERN_TEMPLATE
- #define _GLIBCXX_FAST_MATH
- #define _GLIBCXX_FULLY_DYNAMIC_STRING
- #define _GLIBCXX_HAS_BUILTIN(B)
- #define _GLIBCXX_HAVE__CXA_THREAD_ATEXIT_IMPL
- #define _GLIBCXX_HAVE_ACOSF
- #define _GLIBCXX_HAVE_ACOSL
- #define _GLIBCXX_HAVE_ALIGNED_ALLOC
- #define _GLIBCXX_HAVE_ARC4RANDOM
- #define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE
- #define _GLIBCXX_HAVE_ASINF
- #define _GLIBCXX_HAVE_ASINL
- #define _GLIBCXX_HAVE_AT_QUICK_EXIT
- #define _GLIBCXX_HAVE_ATAN2F
- #define _GLIBCXX_HAVE_ATAN2L
- #define _GLIBCXX_HAVE_ATANF
- #define _GLIBCXX_HAVE_ATANL
- #define _GLIBCXX_HAVE_ATOMIC_LOCK_POLICY
- #define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY
- #define _GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP
- #define _GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE
- #define _GLIBCXX_HAVE_BUILTIN_LAUNDER
- #define _GLIBCXX_HAVE_C99_FLT_EVAL_TYPES
- #define _GLIBCXX_HAVE_CEILF
- #define _GLIBCXX_HAVE_CEILL
- #define _GLIBCXX_HAVE_COSF
- #define _GLIBCXX_HAVE_COSHF
- #define _GLIBCXX_HAVE_COSHL
- #define _GLIBCXX_HAVE_COSL
- #define _GLIBCXX_HAVE_DECL_STRNLEN
- #define _GLIBCXX_HAVE_DIRFD
- #define _GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46
- #define _GLIBCXX_HAVE_EXPF
- #define _GLIBCXX_HAVE_EXPL
- #define _GLIBCXX_HAVE_FABSF
- #define _GLIBCXX_HAVE_FABSL
- #define _GLIBCXX_HAVE_FDOPENDIR

- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_FWRITE_UNLOCKED`
- `#define _GLIBCXX_HAVE_GETENTROPY`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_GETS`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONv`
- `#define _GLIBCXX_HAVE_IS_CONSTANT_EVALUATED`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINK`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_LSEEK`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMALIGN`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_O_NONBLOCK`
- `#define _GLIBCXX_HAVE_OPENAT`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POSIX_MEMALIGN`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`

- `#define _GLIBCXX_HAVE_READLINK`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SECURE_GETENV`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SOCKETATMARK`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STACKTRACE`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMLINK`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TIMESPEC_GET`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_TRUNCATE`
- `#define _GLIBCXX_HAVE_UNLINKAT`
- `#define _GLIBCXX_HAVE_USELOCALE`
- `#define _GLIBCXX_HAVE_VFWSCANF`
- `#define _GLIBCXX_HAVE_VSWSCANF`
- `#define _GLIBCXX_HAVE_VWSCANF`
- `#define _GLIBCXX_HAVE_WCSTOF`
- `#define _GLIBCXX_HAVE_WRITEV`
- `#define _GLIBCXX_HOSTED`
- `#define _GLIBCXX_ICONV_CONST`
- `#define _GLIBCXX_INLINE_VERSION`
- `#define _GLIBCXX_LT_OBJDIR`
- `#define _GLIBCXX_MANGLE_SIZE_T`
- `#define _GLIBCXX_NAMESPACE_CXX11`
- `#define _GLIBCXX_NAMESPACE_LDBL`
- `#define _GLIBCXX_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_NODISCARD`
- `#define _GLIBCXX_NOEXCEPT_PARM`
- `#define _GLIBCXX_NOEXCEPT_QUAL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_RELEASE`
- `#define _GLIBCXX_RES_LIMITS`

- #define _GLIBCXX_STATIC_TZDATA
- #define _GLIBCXX_STD_A
- #define _GLIBCXX_STD_C
- #define _GLIBCXX_STDIO_EOF
- #define _GLIBCXX_STDIO_SEEK_CUR
- #define _GLIBCXX_STDIO_SEEK_END
- #define _GLIBCXX_SYMVER
- #define _GLIBCXX_SYMVER_GNU
- #define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)
- #define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)
- #define _GLIBCXX_THROW_OR_ABORT(_EXC)
- #define _GLIBCXX_TXN_SAFE
- #define _GLIBCXX_TXN_SAFE_DYN
- #define _GLIBCXX_USE_ALLOCATOR_NEW
- #define _GLIBCXX_USE_BUILTIN_TRAIT(BT)
- #define _GLIBCXX_USE_C11_UCHAR_CXX11
- #define _GLIBCXX_USE_C99
- #define _GLIBCXX_USE_C99_COMPLEX
- #define _GLIBCXX_USE_C99_COMPLEX_TR1
- #define _GLIBCXX_USE_C99_CTYPE
- #define _GLIBCXX_USE_C99_CTYPE_TR1
- #define _GLIBCXX_USE_C99_FENV
- #define _GLIBCXX_USE_C99_FENV_TR1
- #define _GLIBCXX_USE_C99_INTTYPES
- #define _GLIBCXX_USE_C99_INTTYPES_TR1
- #define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T
- #define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1
- #define _GLIBCXX_USE_C99_MATH
- #define _GLIBCXX_USE_C99_MATH_FUNCS
- #define _GLIBCXX_USE_C99_MATH_TR1
- #define _GLIBCXX_USE_C99_STDINT
- #define _GLIBCXX_USE_C99_STDINT_TR1
- #define _GLIBCXX_USE_C99_STDIO
- #define _GLIBCXX_USE_C99_STDLIB
- #define _GLIBCXX_USE_C99_WCHAR
- #define _GLIBCXX_USE_CHDIR
- #define _GLIBCXX_USE_CHMOD
- #define _GLIBCXX_USE_CLOCK_MONOTONIC
- #define _GLIBCXX_USE_CLOCK_REALTIME
- #define _GLIBCXX_USE_CXX11_ABI
- #define _GLIBCXX_USE_DECIMAL_FLOAT
- #define _GLIBCXX_USE_DEV_RANDOM
- #define _GLIBCXX_USE_DUAL_ABI
- #define _GLIBCXX_USE_FCHMOD
- #define _GLIBCXX_USE_FCHMODAT
- #define _GLIBCXX_USE_FSEEKO_FTELLO
- #define _GLIBCXX_USE_GET_NPROCS
- #define _GLIBCXX_USE_GETCWD
- #define _GLIBCXX_USE_GETTIMEOFDAY
- #define _GLIBCXX_USE_GLIBC_STDIO_EXT
- #define _GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE

- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_LSTAT`
- `#define _GLIBCXX_USE_MKDIR`
- `#define _GLIBCXX_USE_NANOSLEEP`
- `#define _GLIBCXX_USE_NL_LANGINFO_L`
- `#define _GLIBCXX_USE_NLS`
- `#define _GLIBCXX_USE_PROC_SELF_STATUS`
- `#define _GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT`
- `#define _GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK`
- `#define _GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_REALPATH`
- `#define _GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- `#define _GLIBCXX_USE_SCHED_YIELD`
- `#define _GLIBCXX_USE_SENDFILE`
- `#define _GLIBCXX_USE_ST_MTIM`
- `#define _GLIBCXX_USE_STD_SPEC_FUNCS`
- `#define _GLIBCXX_USE_STDIO_LOCKING`
- `#define _GLIBCXX_USE_STRUCT_TM_TM_ZONE`
- `#define _GLIBCXX_USE_TMPNAM`
- `#define _GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20`
- `#define _GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T`
- `#define _GLIBCXX_USE_UTILITY`
- `#define _GLIBCXX_USE_UTILITY`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_USE_WEAK_REF`
- `#define _GLIBCXX_VERBOSE`
- `#define _GLIBCXX_VERBOSE_ASSERT`
- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define _GLIBCXX_X86_RDRAND`
- `#define _GLIBCXX_X86_RDSEED`
- `#define _GLIBCXX_ZONEINFO_DIR`
- `#define _GTHREAD_USE_MUTEX_TIMEDLOCK`

Typedefs

- `typedef decltype(nullptr) std::nullptr_t`
- `typedef __PTRDIFF_TYPE__ std::ptrdiff_t`
- `typedef __SIZE_TYPE__ std::size_t`

Functions

- `void std::__glibcxx_assert_fail (const char *__file, int __line, const char *__function, const char *__condition) noexcept`
- `constexpr bool std::__is_constant_evaluated () noexcept`
- `void std::__terminate () noexcept`

7.721.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

7.722 `c++io.h` File Reference

Namespaces

- namespace `std`

Typedefs

- typedef FILE `std::__c_file`
- typedef `__pthread_mutex_t` `std::__c_lock`

7.722.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

7.723 `c++locale.h` File Reference

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_C_LOCALE_GNU`
- `#define _GLIBCXX_NUM_CATEGORIES`

Typedefs

- typedef `__locale_t` `std::__c_locale`

Functions

- int `std::__convert_from_v` (const `__c_locale` & `__cloc`, char * `__out`, const int `__size`, const char * `__fmt`,...)

7.723.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.724 `c++locale_internal.h` File Reference

Namespaces

- namespace `std`

Functions

- Catalogs & `std::get_catalogs` ()

7.724.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.725 `parallel/compatibility.h` File Reference

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Tp>`
`_Tp __gnu_parallel::__add_omp` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `template<typename _Tp>`
`bool __gnu_parallel::__cas_omp` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- `template<typename _Tp>`
`bool __gnu_parallel::__compare_and_swap` (volatile `_Tp *``__ptr`, `_Tp` `__comparand`, `_Tp` `__replacement`)
- `template<typename _Tp>`
`_Tp __gnu_parallel::__fetch_and_add` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `void __gnu_parallel::__yield` ()

7.725.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

7.726 `x86_64-pc-linux-gnu/bits/compatibility.h` File Reference

7.726.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

7.727 `cpu_defines.h` File Reference

7.727.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

7.728 `ctype_base.h` File Reference

Classes

- struct [std::ctype_base](#)

Namespaces

- namespace [std](#)

7.728.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.729 `ctype_inline.h` File Reference

Namespaces

- namespace [std](#)

7.729.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.730 cxxabi_tweaks.h File Reference

Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

Typedefs

- `typedef void __cxxabiv1::__cxa_cdtor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

Variables

- `__extension__ typedef int __cxxabiv1::__guard`

7.730.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

7.731 error_constants.h File Reference

Namespaces

- namespace [std](#)

Enumerations

- enum class `errc` {
`address_family_not_supported`, `address_in_use`, `address_not_available`, `already_connected`,
`argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`,
`broken_pipe`, `connection_aborted`, `connection_already_in_progress`, `connection_refused`,
`connection_reset`, `cross_device_link`, `destination_address_required`, `device_or_resource_busy`,
`directory_not_empty`, `executable_format_error`, `file_exists`, `file_too_large`,
`filename_too_long`, `function_not_supported`, `host_unreachable`, `illegal_byte_sequence`,
`inappropriate_io_control_operation`, `interrupted`, `invalid_argument`, `invalid_seek`,
`io_error`, `is_a_directory`, `message_size`, `network_down`,
`network_reset`, `network_unreachable`, `no_buffer_space`, `no_child_process`,
`no_lock_available`, `no_message`, `no_protocol_option`, `no_space_on_device`,
`no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`, `no_such_process`,
`not_a_directory`, `not_a_socket`, `not_connected`, `not_enough_memory`,
`operation_in_progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`,
`permission_denied`, `protocol_not_supported`, `read_only_file_system`, `resource_deadlock_would_occur`,
`resource_unavailable_try_again`, `result_out_of_range`, `timed_out`, `too_many_files_open_in_system`,
`too_many_files_open`, `too_many_links`, `too_many_symbolic_link_levels`, `wrong_protocol_type` }

7.731.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

7.732 extc++.h File Reference

7.732.1 Detailed Description

This is an implementation file for a precompiled header.

7.733 messages_members.h File Reference

Namespaces

- namespace [std](#)

7.733.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.734 opt_random.h File Reference

Namespaces

- namespace [std](#)

7.734.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

7.735 os_defines.h File Reference

Macros

- `#define __NO_CTYPE`
- `#define _GLIBCXX_MAY_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC`

7.735.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

7.736 stdc++.h File Reference

7.736.1 Detailed Description

This is an implementation file for a precompiled header.

7.737 stdtr1c++.h File Reference

7.737.1 Detailed Description

This is an implementation file for a precompiled header.

7.738 time_members.h File Reference

Namespaces

- namespace [std](#)

7.738.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

7.739 compare File Reference

Classes

- struct [std::compare_three_way_result< _Tp, _Up >](#)
- struct [std::type_order< _Tp, _Up >](#)

Namespaces

- namespace [std](#)
- namespace [std::__detail](#)

Typedefs

- `template<typename _Tp, typename _Up>`
using [std::__detail::__cmp3way_res_t](#)
- `template<typename... _Ts>`
using [std::common_comparison_category_t](#)
- `template<typename _Tp, typename _Up = _Tp>`
using [std::compare_three_way_result_t](#)
- using [std::__cmp_cat::type](#)

Enumerations

- enum class [_Ord](#) : type { [equivalent](#) , [less](#) , [greater](#) , [unordered](#) }

Functions

- `template<typename... _Ts>`
constexpr auto [std::__detail::__common_cmp_cat](#) ()
- `template<typename _Ordering>`
constexpr [_Ordering](#) [std::__cmp_cat::make](#) ([_Ord](#) __o) noexcept
- `template<typename _Ordering>`
constexpr [_Ord](#) [std::__cmp_cat::ord](#) ([_Ordering](#) __o) noexcept
- constexpr bool [std::is_eq](#) ([partial_ordering](#) __cmp) noexcept
- constexpr bool [std::is_gt](#) ([partial_ordering](#) __cmp) noexcept
- constexpr bool [std::is_gteq](#) ([partial_ordering](#) __cmp) noexcept
- constexpr bool [std::is_lt](#) ([partial_ordering](#) __cmp) noexcept
- constexpr bool [std::is_lteq](#) ([partial_ordering](#) __cmp) noexcept
- constexpr bool [std::is_neq](#) ([partial_ordering](#) __cmp) noexcept

Variables

- `template<typename _Tp>`
`constexpr unsigned std::__detail::__cmp_cat_id`
- `template<> constexpr unsigned std::__detail::__cmp_cat_id< partial_ordering >`
- `template<> constexpr unsigned std::__detail::__cmp_cat_id< strong_ordering >`
- `template<> constexpr unsigned std::__detail::__cmp_cat_id< weak_ordering >`
- `constexpr __compare::__Partial_fallback std::compare_partial_order_fallback`
- `constexpr __compare::__Strong_fallback std::compare_strong_order_fallback`
- `constexpr __compare::__Weak_fallback std::compare_weak_order_fallback`
- `constexpr __compare::__Partial_order std::partial_order`
- `constexpr __compare::__Strong_order std::strong_order`
- `template<typename _Tp, typename _Up>`
`constexpr strong_ordering std::type_order_v`
- `constexpr __compare::__Weak_order std::weak_order`

7.739.1 Detailed Description

This is a Standard C++ Library header.

7.740 cxxabi.h File Reference

Classes

- class `__gnu_cxx::recursive_init_error`

Namespaces

- namespace `__gnu_cxx`
- namespace `abi`

Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type) (void *)`

Functions

- `__cxa_dependent_exception * __cxxabiv1::__cxa_allocate_dependent_exception () noexcept`
- `int __cxxabiv1::__cxa_atexit (void*)(void *), void *, void *) noexcept`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `void * __cxxabiv1::__cxa_begin_catch (void *) noexcept`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () noexcept`
- `void __cxxabiv1::__cxa_deleted_virtual (void)`
- `char * __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`
- `void __cxxabiv1::__cxa_end_catch ()`
- `void __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_free_dependent_exception (__cxa_dependent_exception *) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `void * __cxxabiv1::__cxa_get_exception_ptr (void *) noexcept`
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals () noexcept`
- `__cxa_eh_globals * __cxxabiv1::__cxa_get_globals_fast () noexcept`
- `void __cxxabiv1::__cxa_guard_abort (__guard *) noexcept`

- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) noexcept`
- `void __cxxabiv1::__cxa_pure_virtual (void)`
- `void __cxxabiv1::__cxa_rethrow ()`
- `int __cxxabiv1::__cxa_thread_atexit (void(*) (void *), void *, void *) noexcept`
- `void __cxxabiv1::__cxa_throw (void *, std::type_info *, void(*) (void *))`
- `void __cxxabiv1::__cxa_throw_bad_array_new_length ()`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __dest_array, void * __src_array, size_t __element_count, size_t __element_size, __cxa_ctor_return_type(*) __constructor)(void *, void *), __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_cleanup (void * __array_address, size_t __element_count, size_t __s, __cxa_ctor_type __destructor) noexcept`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete2 (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(*) __dealloc)(void *)`
- `void __cxxabiv1::__cxa_vec_delete3 (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(*) __dealloc)(void *, size_t)`
- `void __cxxabiv1::__cxa_vec_dtor (void * __array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new2 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void(*) __alloc)(size_t), void(*) __dealloc)(void *)`
- `void * __cxxabiv1::__cxa_vec_new3 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void(*) __alloc)(size_t), void(*) __dealloc)(void *, size_t)`
- `void * __cxxabiv1::__dynamic_cast (const void * __src_ptr, const __class_type_info * __src_type, const __class_type_info * __dst_type, ptrdiff_t __src2dst)`

7.740.1 Detailed Description

The header provides an interface to the C++ ABI.

7.740.2 Function Documentation

`__cxa_demangle()`

```
char * __cxxabiv1::__cxa_demangle (
    const char * __mangled_name,
    char * __output_buffer,
    size_t * __length,
    int * __status)
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

Parameters

<code>__mangled_name</code>	A NUL-terminated character string containing the name to be demangled.
-----------------------------	--

<code>__output_buffer</code>	A region of memory, allocated with <code>malloc</code> , of <code>*__length</code> bytes, into which the demangled name is stored. If <code>__output_buffer</code> is not long enough, it is expanded using <code>realloc</code> . <code>__output_buffer</code> may instead be null; in that case, the demangled name is placed in a region of memory allocated with <code>malloc</code> .
<code>__length</code>	If <code>__length</code> is non-null, the length of the buffer containing the demangled name is placed in <code>*__length</code> .
<code>__status</code>	If <code>__status</code> is non-null, <code>*__status</code> is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: <i>mangled_name</i> is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

Returns

A pointer to the start of the NUL-terminated demangled name, or a null pointer if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in `__gnu_cxx::__verbose_terminate_handler`.

See https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html for other examples of use.

Note

The same demangling functionality is available via `libiberty` (`<libiberty/demangle.h>` and `libiberty.a`) in GCC 3.1 and later, but that requires explicit installation (`-enable-install-libiberty`) and uses a different API, although the ABI is unchanged.

7.741 exception File Reference

Classes

- class `std::bad_exception`

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Typedefs

- typedef `void(* std::terminate_handler)()`

Functions

- `void __gnu_cxx::__verbose_terminate_handler()`
- `terminate_handler std::get_terminate() noexcept`
- `terminate_handler std::set_terminate(terminate_handler) noexcept`
- `void std::terminate() noexcept`
- `bool std::uncaught_exception() noexcept`

7.741.1 Detailed Description

This is a Standard C++ Library header.

7.742 `initializer_list` File Reference

Classes

- class `std::initializer_list<_E>`

Namespaces

- namespace `std`

7.742.1 Detailed Description

This is a Standard C++ Library header.

7.743 `new` File Reference

Classes

- struct `std::destroying_delete_t`

Namespaces

- namespace `std`

Macros

- `#define _GLIBCXX_PLACEMENT_CONSTEXPR`

Typedefs

- typedef void(* `std::new_handler`) ()

Enumerations

- enum class `align_val_t` : `size_t`

Functions

- `new_handler` `std::get_new_handler` () noexcept
- void **operator delete** (void *) noexcept
- void **operator delete** (void *, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::align_val_t) noexcept
- void **operator delete** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::size_t) noexcept
- void **operator delete** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete** (void *, void *) noexcept
- void **operator delete[]** (void *) noexcept
- void **operator delete[]** (void *, const std::nothrow_t &) noexcept
- void **operator delete[]** (void *, std::align_val_t) noexcept
- void **operator delete[]** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete[]** (void *, std::size_t) noexcept
- void **operator delete[]** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete[]** (void *, void *) noexcept
- void * **operator new** (std::size_t)
- void * **operator new** (std::size_t, const std::nothrow_t &) noexcept

- void * **operator new** (std::size_t, std::align_val_t)
- void * **operator new** (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new** (std::size_t, void *__p) noexcept
- void * **operator new**[] (std::size_t)
- void * **operator new**[] (std::size_t, const std::nothrow_t &) noexcept
- void * **operator new**[] (std::size_t, std::align_val_t)
- void * **operator new**[] (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new**[] (std::size_t, void *__p) noexcept
- [new_handler](#) **std::set_new_handler** ([new_handler](#)) throw ()

Variables

- constexpr [destroying_delete_t](#) **std::destroying_delete**
- const nothrow_t **std::nothrow**

7.743.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic_memory.html for more.

7.743.2 Function Documentation

operator new()

```
void * operator new (
    std::size_t ) [nodiscard]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

7.744 typeinfo File Reference

Classes

- class [std::bad_cast](#)
- class [std::bad_typeid](#)
- class [std::type_info](#)

Namespaces

- namespace [std](#)

Macros

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`

7.744.1 Detailed Description

This is a Standard C++ Library header.

Index

- `_AlgorithmStrategy`
 - `__gnu_parallel`, [432](#)
- `_BALLOC_ALIGN_BYTES`
 - `bitmap_allocator.h`, [3545](#)
- `_BinIndex`
 - `__gnu_parallel`, [432](#)
- `_Bit_scan_forward`
 - `__gnu_cxx`, [404](#)
- `_CASable`
 - `__gnu_parallel`, [432](#)
- `_CASable_bits`
 - `__gnu_parallel`, [473](#)
- `_CASable_mask`
 - `__gnu_parallel`, [473](#)
- `_Construct`
 - `std`, [598](#)
- `_DRandomShufflingGlobalData`
 - `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter >`, [901](#)
- `_Destroy`
 - `std`, [598](#), [599](#)
- `_Destroy_n`
 - `std`, [599](#)
- `_Distance_precision`
 - `__gnu_debug`, [422](#)
- `_FindAlgorithm`
 - `__gnu_parallel`, [432](#)
- `_Find_first`
 - `SGI`, [157](#)
- `_Find_next`
 - `SGI`, [157](#)
- `_GLIBCXX_BAL_QUICKSORT`
 - `features.h`, [3665](#)
- `_GLIBCXX_CALL`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_DEBUG_VERIFY_AT_F`
 - `macros.h`, [3521](#)
- `_GLIBCXX_DEQUE_BUF_SIZE`
 - `stl_deque.h`, [3423](#)
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
 - `features.h`, [3665](#)
- `_GLIBCXX_FIND_EQUAL_SPLIT`
 - `features.h`, [3666](#)
- `_GLIBCXX_FIND_GROWING_BLOCKS`
 - `features.h`, [3666](#)
- `_GLIBCXX_MERGESORT`
 - `features.h`, [3666](#)
- `_GLIBCXX_PARALLEL_ASSERTIONS`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_PARALLEL_CONDITION`
 - `settings.h`, [3682](#)
- `_GLIBCXX_PARALLEL_LENGTH`
 - `multiway_merge.h`, [3674](#)
- `_GLIBCXX_QUICKSORT`
 - `features.h`, [3666](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_SCALE_DOWN_FPU`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
 - `features.h`, [3666](#)
- `_GLIBCXX_TREE_FULL_COPY`
 - `features.h`, [3666](#)
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
 - `features.h`, [3667](#)
- `_GLIBCXX_VERBOSE_LEVEL`
 - `compiletime_settings.h`, [3664](#)
- `_GLIBCXX_VOLATILE`
 - `partition.h`, [3678](#)
 - `queue.h`, [3679](#)
- `_GuardedIterator`
 - `__gnu_parallel::_GuardedIterator<_RAIter, _Compare >`, [915](#)
- `_LoserTreeBase`
 - `__gnu_parallel::_LoserTreeBase<_Tp, _Compare >`, [948](#)
- `_M_allocate_and_copy`
 - `std::vector<_Tp, _Alloc >`, [3240](#)
 - `std::vector<bool, _Alloc >`, [3257](#)
- `_M_allocate_single_object`
 - `__gnu_cxx::bitmap_allocator<_Tp >`, [1808](#)
- `_M_attach`
 - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [993](#)
 - `__gnu_debug::_Safe_iterator_base`, [1002](#)
 - `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer >`, [1008](#)
 - `__gnu_debug::_Safe_local_iterator_base`, [1016](#)
- `_M_attach_single`
 - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [994](#)
 - `__gnu_debug::_Safe_iterator_base`, [1002](#)
 - `__gnu_debug::_Safe_local_iterator<_Iterator, _UContainer >`, [1008](#), [1009](#)
 - `__gnu_debug::_Safe_local_iterator_base`, [1016](#)
- `_M_attached_to`
 - `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category >`, [994](#)
 - `__gnu_debug::_Safe_iterator_base`, [1002](#)

- `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
- `__gnu_debug::Safe_local_iterator_base`, 1016
- `_M_before_dereferenceable`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 994
- `_M_begin`
 - `__gnu_parallel::Piece< _DifferenceTp >`, 971
- `_M_bin_proc`
 - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 901
- `_M_bins_begin`
 - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 903
- `_M_buf`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2135
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2964
 - `std::basic_filebuf< _CharT, _Traits >`, 1139
- `_M_buf_locale`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2135
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2964
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2982
 - `std::basic_filebuf< _CharT, _Traits >`, 1139
 - `std::basic_streambuf< _CharT, _Traits >`, 1511
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1719
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3274
- `_M_buf_size`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2135
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2964
 - `std::basic_filebuf< _CharT, _Traits >`, 1139
- `_M_can_compare`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 994
 - `__gnu_debug::Safe_iterator_base`, 1002
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
 - `__gnu_debug::Safe_local_iterator_base`, 1016
- `_M_clear`
 - `__gnu_cxx::free_list`, 2199
- `_M_comp`
 - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, 949
- `_M_const_iterators`
 - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 989
 - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1021
 - `__gnu_debug::Safe_sequence< _Sequence >`, 1024
 - `__gnu_debug::Safe_sequence_base`, 1027
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1030
 - `__gnu_debug::Safe_unordered_container_base`, 1033
- `_M_const_local_iterators`
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1030
 - `__gnu_debug::Safe_unordered_container_base`, 1033
- `_M_create_node`
 - `std::list< _Tp, _Alloc >`, 2372
- `_M_create_pback`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2123
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2948
 - `std::basic_filebuf< _CharT, _Traits >`, 1123
- `_M_deallocate_single_object`
 - `__gnu_cxx::bitmap_allocator< _Tp >`, 1808
- `_M_dereferenceable`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 994
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
- `_M_destroy_pback`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2123
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2948
 - `std::basic_filebuf< _CharT, _Traits >`, 1123
- `_M_detach`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
 - `__gnu_debug::Safe_iterator_base`, 1002
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
 - `__gnu_debug::Safe_local_iterator_base`, 1017
- `_M_detach_all`
 - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 989
 - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1020
 - `__gnu_debug::Safe_sequence< _Sequence >`, 1023
 - `__gnu_debug::Safe_sequence_base`, 1026
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1029
 - `__gnu_debug::Safe_unordered_container_base`, 1032
- `_M_detach_single`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
 - `__gnu_debug::Safe_iterator_base`, 1002
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
 - `__gnu_debug::Safe_local_iterator_base`, 1017
- `_M_detach_singular`
 - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 989
 - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1020

- __gnu_debug::Safe_sequence< _Sequence >, 1023
- __gnu_debug::Safe_sequence_base, 1026
- __gnu_debug::Safe_unordered_container< _Container >, 1029
- __gnu_debug::Safe_unordered_container_base, 1032
- _M_dist
 - __gnu_parallel::DRandomShufflingGlobalData< _RAIter >, 901
- _M_elements_leftover
 - __gnu_parallel::QSBThreadLocal< _RAIter >, 980
- _M_end
 - __gnu_parallel::Piece< _DifferenceTp >, 971
- _M_ext_buf
 - __gnu_cxx::enc_filebuf< _CharT >, 2136
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2964
 - std::basic_filebuf< _CharT, _Traits >, 1139
- _M_ext_buf_size
 - __gnu_cxx::enc_filebuf< _CharT >, 2136
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2964
 - std::basic_filebuf< _CharT, _Traits >, 1139
- _M_ext_next
 - __gnu_cxx::enc_filebuf< _CharT >, 2136
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2965
 - std::basic_filebuf< _CharT, _Traits >, 1139
- _M_fill_initialize
 - std::deque< _Tp, _Alloc >, 2071
- _M_finish_iterator
 - __gnu_parallel::__accumulate_selector< _It >, 754
 - __gnu_parallel::__adjacent_difference_selector< _It >, 755
 - __gnu_parallel::__count_if_selector< _It, _Diff >, 773
 - __gnu_parallel::__count_selector< _It, _Diff >, 774
 - __gnu_parallel::__fill_selector< _It >, 791
 - __gnu_parallel::__for_each_selector< _It >, 796
 - __gnu_parallel::__generate_selector< _It >, 797
 - __gnu_parallel::__generic_for_each_selector< _It >, 800
 - __gnu_parallel::__identity_selector< _It >, 801
 - __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 803
 - __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 828
 - __gnu_parallel::__replace_selector< _It, _Tp >, 830
 - __gnu_parallel::__transform1_selector< _It >, 832
 - __gnu_parallel::__transform2_selector< _It >, 833
- _M_first
 - __gnu_parallel::__Job< _DifferenceTp >, 930
- _M_first_insert
 - __gnu_parallel::__LoserTreeBase< _Tp, _Compare >, 949
- _M_gcount
 - std::basic_fstream< _CharT, _Traits >, 1186
 - std::basic_ifstream< _CharT, _Traits >, 1230
 - std::basic_iostream< _CharT, _Traits >, 1301
 - std::basic_istream< _CharT, _Traits >, 1340
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1381
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1764
- _M_get
 - __gnu_cxx::free_list, 2199
- _M_get_mutex
 - __gnu_debug::Safe_forward_list< _SafeSequence >, 989
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, 995
 - __gnu_debug::Safe_iterator_base, 1003
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, 1009
 - __gnu_debug::Safe_local_iterator_base, 1017
 - __gnu_debug::Safe_node_sequence< _Sequence >, 1020
 - __gnu_debug::Safe_sequence< _Sequence >, 1023
 - __gnu_debug::Safe_sequence_base, 1026
 - __gnu_debug::Safe_unordered_container< _Container >, 1029
 - __gnu_debug::Safe_unordered_container_base, 1032
- _M_get_result
 - std::__basic_future< _Res >, 765
 - std::future< _Res >, 2212
 - std::future< _Res & >, 2213
 - std::future< void >, 2215
 - std::shared_future< _Res >, 2909
 - std::shared_future< _Res & >, 2910
 - std::shared_future< void >, 2912
- _M_getloc
 - std::basic_fstream< _CharT, _Traits >, 1151
 - std::basic_ifstream< _CharT, _Traits >, 1202
 - std::basic_ios< _CharT, _Traits >, 1243
 - std::basic_iostream< _CharT, _Traits >, 1267
 - std::basic_istream< _CharT, _Traits >, 1314
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1354
 - std::basic_ofstream< _CharT, _Traits >, 1394
 - std::basic_ostream< _CharT, _Traits >, 1427
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1460
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1729
 - std::ios_base, 2300
- _M_global
 - __gnu_parallel::QSBThreadLocal< _RAIter >, 980
- _M_in_beg

- `__gnu_cxx::enc_filebuf< _CharT >`, 2136
- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2965
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2982
- `std::basic_filebuf< _CharT, _Traits >`, 1139
- `std::basic_streambuf< _CharT, _Traits >`, 1511
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1719
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3275
- `_M_in_cur`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2136
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2965
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2982
 - `std::basic_filebuf< _CharT, _Traits >`, 1139
 - `std::basic_streambuf< _CharT, _Traits >`, 1511
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1719
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3275
- `_M_in_end`
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2136
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2965
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2982
 - `std::basic_filebuf< _CharT, _Traits >`, 1140
 - `std::basic_streambuf< _CharT, _Traits >`, 1511
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1719
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3275
- `_M_in_same_bucket`
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1009
- `_M_incrementable`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1010
- `_M_initial`
 - `__gnu_parallel::QSBThreadLocal< _RAIter >`, 980
- `_M_initialize_map`
 - `std::Deque_base< _Tp, _Alloc >`, 898
- `_M_insert`
 - `__gnu_cxx::free_list`, 2199
- `_M_invalidate`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
 - `__gnu_debug::Safe_iterator_base`, 1003
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1010
 - `__gnu_debug::Safe_local_iterator_base`, 1017
- `_M_invalidate_all`
 - `__gnu_debug::Safe_sequence< _Sequence >`, 1023
 - `__gnu_debug::Safe_sequence_base`, 1027
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1029
 - `__gnu_debug::Safe_unordered_container_base`, 1033
- `_M_invalidate_if`
 - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 989
 - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1020
 - `__gnu_debug::Safe_sequence< _Sequence >`, 1023
- `_M_is_before_begin`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
- `_M_is_begin`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 995
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1010
- `_M_is_beginnest`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 996
- `_M_is_end`
 - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 996
 - `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1010
- `_M_iterators`
 - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 989
 - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1021
 - `__gnu_debug::Safe_sequence< _Sequence >`, 1024
 - `__gnu_debug::Safe_sequence_base`, 1027
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1030
 - `__gnu_debug::Safe_unordered_container_base`, 1033
- `_M_key`
 - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser`, 942
- `_M_last`
 - `__gnu_parallel::Job< _DifferenceTp >`, 930
- `_M_leftover_parts`
 - `__gnu_parallel::QSBThreadLocal< _RAIter >`, 980
- `_M_load`
 - `__gnu_parallel::Job< _DifferenceTp >`, 930
- `_M_local_iterators`
 - `__gnu_debug::Safe_unordered_container< _Container >`, 1030
 - `__gnu_debug::Safe_unordered_container_base`, 1033
- `_M_log_k`
 - `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >`, 944
 - `__gnu_parallel::LoserTree< false, _Tp, _Compare`

- [>](#), [946](#)
- [__gnu_parallel::LoserTreeBase< _Tp, _Compare](#)
[>](#), [949](#)
- [_M_losers](#)
 - [__gnu_parallel::LoserTreeBase< _Tp, _Compare](#)
[>](#), [949](#)
- [_M_mode](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2136](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2965](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
 - [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [1719](#)
- [_M_new_elements_at_back](#)
 - [std::deque< _Tp, _Alloc >](#), [2071](#)
- [_M_new_elements_at_front](#)
 - [std::deque< _Tp, _Alloc >](#), [2071](#)
- [_M_next](#)
 - [__gnu_debug::Safe_iterator< _Iterator, _Se-](#)
[quence, _Category >](#), [998](#)
 - [__gnu_debug::Safe_iterator_base](#), [1003](#)
 - [__gnu_debug::Safe_local_iterator< _Iterator,](#)
[_UContainer >](#), [1013](#)
 - [__gnu_debug::Safe_local_iterator_base](#), [1018](#)
- [_M_num_bins](#)
 - [__gnu_parallel::DRandomShufflingGlobalData<](#)
[_RAIter >](#), [901](#)
- [_M_num_bits](#)
 - [__gnu_parallel::DRandomShufflingGlobalData<](#)
[_RAIter >](#), [901](#)
- [_M_num_threads](#)
 - [__gnu_parallel::DRSSorterPU< _RAIter, _Ran-](#)
[domNumberGenerator >](#), [903](#)
 - [__gnu_parallel::PMWMSSortingData< _RAIter >](#),
[974](#)
 - [__gnu_parallel::QSBThreadLocal< _RAIter >](#), [980](#)
- [_M_offsets](#)
 - [__gnu_parallel::PMWMSSortingData< _RAIter >](#),
[974](#)
- [_M_out_beg](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2136](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2965](#)
 - [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#),
[2982](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
 - [std::basic_streambuf< _CharT, _Traits >](#), [1511](#)
 - [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [1719](#)
 - [std::wbuffer_convert< _Codecv, _Elem, _Tr >](#), [3275](#)
- [_M_out_cur](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2965](#)
 - [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#),
[2982](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
 - [std::basic_streambuf< _CharT, _Traits >](#), [1512](#)
 - [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [1719](#)
- [std::wbuffer_convert< _Codecv, _Elem, _Tr >](#), [3275](#)
- [_M_out_end](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2965](#)
 - [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#),
[2982](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
 - [std::basic_streambuf< _CharT, _Traits >](#), [1512](#)
 - [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [1719](#)
 - [std::wbuffer_convert< _Codecv, _Elem, _Tr >](#), [3275](#)
- [_M_pback](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2965](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
- [_M_pback_cur_save](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2966](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1140](#)
- [_M_pback_end_save](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2966](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1141](#)
- [_M_pback_init](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2966](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1141](#)
- [_M_pieces](#)
 - [__gnu_parallel::PMWMSSortingData< _RAIter >](#),
[974](#)
- [_M_pop_back_aux](#)
 - [std::deque< _Tp, _Alloc >](#), [2071](#)
- [_M_pop_front_aux](#)
 - [std::deque< _Tp, _Alloc >](#), [2071](#)
- [_M_prior](#)
 - [__gnu_debug::Safe_iterator< _Iterator, _Se-](#)
[quence, _Category >](#), [998](#)
 - [__gnu_debug::Safe_iterator_base](#), [1003](#)
 - [__gnu_debug::Safe_local_iterator< _Iterator,](#)
[_UContainer >](#), [1013](#)
 - [__gnu_debug::Safe_local_iterator_base](#), [1018](#)
- [_M_push_back_aux](#)
 - [std::deque< _Tp, _Alloc >](#), [2071](#)
- [_M_push_front_aux](#)
 - [std::deque< _Tp, _Alloc >](#), [2072](#)
- [_M_range_check](#)
 - [std::deque< _Tp, _Alloc >](#), [2072](#)
 - [std::vector< _Tp, _Alloc >](#), [3240](#)
- [_M_range_initialize](#)
 - [std::deque< _Tp, _Alloc >](#), [2072](#)
- [_M_reading](#)
 - [__gnu_cxx::enc_filebuf< _CharT >](#), [2137](#)
 - [__gnu_cxx::stdio_filebuf< _CharT, _Traits >](#), [2966](#)
 - [std::basic_filebuf< _CharT, _Traits >](#), [1141](#)
- [_M_reallocate_map](#)

- std::deque< _Tp, _Alloc >, [2073](#)
- _M_requested_size
 - __gnu_cxx::Temporary_buffer< _ForwardIterator, _Tp >, [1048](#)
 - __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, [3005](#)
 - std::Temporary_buffer< _ForwardIterator, _Tp >, [1050](#)
- _M_reserve_elements_at_back
 - std::deque< _Tp, _Alloc >, [2073](#)
- _M_reserve_elements_at_front
 - std::deque< _Tp, _Alloc >, [2073](#)
- _M_reserve_map_at_back
 - std::deque< _Tp, _Alloc >, [2073](#)
- _M_reserve_map_at_front
 - std::deque< _Tp, _Alloc >, [2073](#)
- _M_reset
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, [996](#)
 - __gnu_debug::Safe_iterator_base, [1003](#)
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, [1010](#)
 - __gnu_debug::Safe_local_iterator_base, [1017](#)
- _M_revalidate_singular
 - __gnu_debug::Safe_forward_list< _SafeSequence >, [989](#)
 - __gnu_debug::Safe_node_sequence< _Sequence >, [1020](#)
 - __gnu_debug::Safe_sequence< _Sequence >, [1023](#)
 - __gnu_debug::Safe_sequence_base, [1027](#)
 - __gnu_debug::Safe_unordered_container< _Container >, [1029](#)
 - __gnu_debug::Safe_unordered_container_base, [1033](#)
- _M_samples
 - __gnu_parallel::PMWMSortingData< _RAIter >, [974](#)
- _M_sd
 - __gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >, [903](#)
- _M_seed
 - __gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >, [903](#)
- _M_sequence
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, [999](#)
 - __gnu_debug::Safe_iterator_base, [1004](#)
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, [1013](#)
 - __gnu_debug::Safe_local_iterator_base, [1018](#)
- _M_sequential_algorithm
 - __gnu_parallel::__adjacent_find_selector, [756](#)
 - __gnu_parallel::__find_first_of_selector< _FIterator >, [792](#)
 - __gnu_parallel::__find_if_selector, [793](#)
 - __gnu_parallel::__mismatch_selector, [806](#)
- _M_set_buffer
 - __gnu_cxx::enc_filebuf< _CharT >, [2123](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2948](#)
 - std::basic_filebuf< _CharT, _Traits >, [1123](#)
- _M_set_node
 - std::Deque_iterator< _Tp, _Ref, _Ptr >, [900](#)
- _M_singular
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, [996](#)
 - __gnu_debug::Safe_iterator_base, [1003](#)
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, [1010](#)
 - __gnu_debug::Safe_local_iterator_base, [1017](#)
- _M_source
 - __gnu_parallel::DRandomShufflingGlobalData< _RAIter >, [902](#)
 - __gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser, [942](#)
 - __gnu_parallel::PMWMSortingData< _RAIter >, [974](#)
- _M_starts
 - __gnu_parallel::DRandomShufflingGlobalData< _RAIter >, [902](#)
 - __gnu_parallel::PMWMSortingData< _RAIter >, [974](#)
- _M_sup
 - __gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser, [942](#)
- _M_swap
 - __gnu_debug::Safe_forward_list< _SafeSequence >, [989](#)
 - __gnu_debug::Safe_node_sequence< _Sequence >, [1020](#)
 - __gnu_debug::Safe_sequence< _Sequence >, [1023](#)
 - __gnu_debug::Safe_sequence_base, [1027](#)
 - __gnu_debug::Safe_unordered_container< _Container >, [1029](#)
 - __gnu_debug::Safe_unordered_container_base, [1033](#)
- _M_temporaries
 - __gnu_parallel::DRandomShufflingGlobalData< _RAIter >, [902](#)
- _M_temporary
 - __gnu_parallel::PMWMSortingData< _RAIter >, [974](#)
- _M_transfer_from_if
 - __gnu_debug::Safe_forward_list< _SafeSequence >, [989](#)
 - __gnu_debug::Safe_node_sequence< _Sequence >, [1020](#)

- `__gnu_debug::__Safe_sequence< _Sequence >`, 1024
- `_M_unlink`
 - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 996
 - `__gnu_debug::__Safe_iterator_base`, 1003
 - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 1010
 - `__gnu_debug::__Safe_local_iterator_base`, 1017
- `_M_use_pointer`
 - `__gnu_parallel::__LoserTreeTraits< _Tp >`, 956
- `_M_value_initialized`
 - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 996
 - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 1011
- `_M_version`
 - `__gnu_debug::__Safe_forward_list< _SafeSequence >`, 990
 - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 999
 - `__gnu_debug::__Safe_iterator_base`, 1004
 - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 1013
 - `__gnu_debug::__Safe_local_iterator_base`, 1018
 - `__gnu_debug::__Safe_node_sequence< _Sequence >`, 1021
 - `__gnu_debug::__Safe_sequence< _Sequence >`, 1024
 - `__gnu_debug::__Safe_sequence_base`, 1027
 - `__gnu_debug::__Safe_unordered_container< _Container >`, 1030
 - `__gnu_debug::__Safe_unordered_container_base`, 1034
- `_M_w`
 - `std::__Base_bitset< 0 >`, 890
 - `std::__Base_bitset< _Nw >`, 889
 - `std::tr2::__dynamic_bitset_base< _WordT, _Alloc >`, 790
- `_MultiwayMergeAlgorithm`
 - `__gnu_parallel`, 432
- `_Opcode`
 - Base and Implementation Classes, 279
- `_Parallelism`
 - `__gnu_parallel`, 432
- `_PartialSumAlgorithm`
 - `__gnu_parallel`, 432
- `_Piece`
 - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 979
- `_PseudoSequence`
 - `__gnu_parallel::__PseudoSequence< _Tp, _DifferenceTp >`, 977
- `_QSBThreadLocal`
 - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 979
- `_RandomNumber`
 - `__gnu_parallel::__RandomNumber`, 981
- `_RestrictedBoundedConcurrentQueue`
 - `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >`, 985
- `_S_constant`
 - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 997
 - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 1011
- `_Safe_iterator`
 - `__gnu_debug::__Safe_iterator< _Iterator, _Sequence, _Category >`, 992, 993
- `_Safe_iterator_base`
 - `__gnu_debug::__Safe_iterator_base`, 1001
- `_Safe_local_iterator`
 - `__gnu_debug::__Safe_local_iterator< _Iterator, _UContainer >`, 1007
- `_Safe_local_iterator_base`
 - `__gnu_debug::__Safe_local_iterator_base`, 1015
- `_SequenceIndex`
 - `__gnu_parallel`, 432
- `_SortAlgorithm`
 - `__gnu_parallel`, 433
- `_SplittingAlgorithm`
 - `__gnu_parallel`, 433
- `_Temporary_buffer`
 - `__gnu_cxx::__Temporary_buffer< _ForwardIterator, _Tp >`, 1048
 - `std::__Temporary_buffer< _ForwardIterator, _Tp >`, 1049
- `_ThreadIndex`
 - `__gnu_parallel`, 432
- `_TokenT`
 - `std::__detail::__Scanner< _CharT >`, 1037
- `_Unchecked_flip`
 - SGI, 157
- `_Unchecked_reset`
 - SGI, 158
- `_Unchecked_set`
 - SGI, 158
- `_Unchecked_test`
 - SGI, 158
- `__addressof`
 - Utilities, 306
- `__allocator_base`
 - Allocators, 334
- `__base`
 - `__gnu_debug`, 422
- `__begin1_iterator`
 - `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`, 803
- `__begin2_iterator`

- __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, [803](#)
- __bins_end
 - __gnu_parallel::__DRSSorterPU< _RAIter, _RandomNumberGenerator >, [902](#)
- __bit_allocate
 - __gnu_cxx::__detail, [415](#)
- __bit_free
 - __gnu_cxx::__detail, [415](#)
- __calc_borders
 - __gnu_parallel, [433](#)
- __check_singular
 - __gnu_debug, [422](#)
- __check_singular_aux
 - __gnu_debug, [422](#)
- __check_string
 - __gnu_debug, [422](#)
- __compare_and_swap
 - __gnu_parallel, [433](#)
- __cpp_lib_experimental_detect
 - Detection idiom, [291](#)
- __ctype_type
 - std::basic_ios< _CharT, _Traits >, [1239](#)
- __cxa_demangle
 - cxxabi.h, [3737](#)
- __cxxabiv1::__forced_unwind, [796](#)
- __decode2
 - __gnu_parallel, [433](#)
- __delete_min_insert
 - __gnu_parallel::__LoserTree< __stable, _Tp, _Compare >, [943](#)
 - __gnu_parallel::__LoserTree< false, _Tp, _Compare >, [945](#)
- __determine_samples
 - __gnu_parallel, [434](#)
- __encode2
 - __gnu_parallel, [434](#)
- __equally_split
 - __gnu_parallel, [435](#)
- __equally_split_point
 - __gnu_parallel, [435](#)
- __fetch_and_add
 - __gnu_parallel, [436](#)
- __find_if_not
 - std, [593](#)
- __find_if_not_n
 - std, [593](#)
- __find_template
 - __gnu_parallel, [436–438](#)
- __for_each_template_random_access
 - __gnu_parallel, [438](#)
- __for_each_template_random_access_ed
 - __gnu_parallel, [439](#)
- __for_each_template_random_access_omp_loop
 - __gnu_parallel, [440](#)
- __for_each_template_random_access_omp_loop_static
 - __gnu_parallel, [441](#)
- __for_each_template_random_access_workstealing
 - __gnu_parallel, [442](#)
- __foreign_iterator_aux2
 - __gnu_debug, [423](#)
- __from_chars_alnum
 - std::__detail, [668](#)
- __from_chars_pow2_base
 - std::__detail, [668](#)
- __gcd
 - std, [593](#)
- __gen_two_uniform_ints
 - std, [593](#)
- __genrand_bits
 - __gnu_parallel::__RandomNumber, [982](#)
- __get_distance
 - __gnu_debug, [423](#)
- __get_min_source
 - __gnu_parallel::__LoserTree< __stable, _Tp, _Compare >, [944](#)
 - __gnu_parallel::__LoserTree< false, _Tp, _Compare >, [946](#)
 - __gnu_parallel::__LoserTreeBase< _Tp, _Compare >, [948](#)
- __get_num_threads
 - __gnu_parallel::balanced_quicksort_tag, [1116](#)
 - __gnu_parallel::balanced_tag, [1117](#)
 - __gnu_parallel::default_parallel_tag, [2058](#)
 - __gnu_parallel::exact_tag, [2153](#)
 - __gnu_parallel::multiway_mergesort_exact_tag, [2556](#)
 - __gnu_parallel::multiway_mergesort_sampling_tag, [2557](#)
 - __gnu_parallel::multiway_mergesort_tag, [2558](#)
 - __gnu_parallel::omp_loop_static_tag, [2693](#)
 - __gnu_parallel::omp_loop_tag, [2694](#)
 - __gnu_parallel::parallel_tag, [2734](#)
 - __gnu_parallel::quicksort_tag, [2797](#)
 - __gnu_parallel::sampling_tag, [2877](#)
 - __gnu_parallel::unbalanced_tag, [3098](#)
- __glibcxx_check_erase
 - macros.h, [3519](#)
- __glibcxx_check_erase_after
 - macros.h, [3519](#)
- __glibcxx_check_erase_range
 - macros.h, [3520](#)
- __glibcxx_check_erase_range_after
 - macros.h, [3520](#)
- __glibcxx_check_heap_pred
 - macros.h, [3520](#)
- __glibcxx_check_insert
 - macros.h, [3520](#)

- `__glibcxx_check_insert_after`
 macros.h, [3520](#)
- `__glibcxx_check_insert_range`
 macros.h, [3520](#)
- `__glibcxx_check_insert_range_after`
 macros.h, [3521](#)
- `__glibcxx_check_partitioned_lower`
 macros.h, [3521](#)
- `__glibcxx_check_partitioned_lower_pred`
 macros.h, [3521](#)
- `__glibcxx_check_partitioned_upper_pred`
 macros.h, [3521](#)
- `__glibcxx_check_sorted_pred`
 macros.h, [3521](#)
- `__gnu_cxx`, [387](#)
 - `__Bit_scan_forward`, [404](#)
 - `__int_traits`, [404](#)
 - `__static_pointer_cast`, [404](#)
 - `iota`, [404](#)
 - `operator!=`, [405](#)
 - `operator<`, [408](#)
 - `operator<=`, [409](#), [410](#)
 - `operator>`, [411](#), [412](#)
 - `operator>=`, [412](#), [413](#)
 - `operator+`, [406](#), [407](#)
 - `operator==`, [410](#), [411](#)
 - `swap`, [414](#)
- `__gnu_cxx::Caster<_ToType>`, [893](#)
- `__gnu_cxx::Char_types<_CharT>`, [894](#)
- `__gnu_cxx::ExtPtr_allocator<_Tp>`, [907](#)
- `__gnu_cxx::Invalid_type`, [920](#)
- `__gnu_cxx::Pointer_adapter<_Storage_policy>`, [975](#)
- `__gnu_cxx::Relative_pointer_impl<_Tp>`, [982](#)
- `__gnu_cxx::Relative_pointer_impl<const _Tp>`, [984](#)
- `__gnu_cxx::Std_pointer_impl<_Tp>`, [1046](#)
- `__gnu_cxx::Temporary_buffer<_ForwardIterator, _Tp>`, [1047](#)
 - `_M_requested_size`, [1048](#)
 - `_Temporary_buffer`, [1048](#)
 - `begin`, [1048](#)
 - `end`, [1048](#)
 - `size`, [1048](#)
- `__gnu_cxx::Unqualified_type<_Tp>`, [1050](#)
- `__gnu_cxx::__alloc_traits<_Alloc, typename>`, [756](#)
 - `allocate`, [759](#), [760](#)
 - `allocate_at_least`, [760](#)
 - `const_void_pointer`, [758](#)
 - `construct`, [761](#)
 - `deallocate`, [761](#), [762](#)
 - `destroy`, [762](#)
 - `is_always_equal`, [758](#)
 - `max_size`, [762](#)
 - `propagate_on_container_copy_assignment`, [758](#)
 - `propagate_on_container_move_assignment`, [758](#)
 - `propagate_on_container_swap`, [759](#)
 - `select_on_container_copy_construction`, [763](#)
 - `void_pointer`, [759](#)
- `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>`, [771](#)
- `__gnu_cxx::__detail`, [414](#)
 - `__bit_allocate`, [415](#)
 - `__bit_free`, [415](#)
 - `__num_bitmaps`, [415](#)
 - `__num_blocks`, [415](#)
- `__gnu_cxx::__detail::Bitmap_counter<_Tp>`, [892](#)
- `__gnu_cxx::__detail::Ffit_finder<_Tp>`, [908](#)
- `__gnu_cxx::__detail::__mini_vector<_Tp>`, [805](#)
- `__gnu_cxx::mt_alloc<_Tp, _Poolp>`, [807](#)
- `__gnu_cxx::mt_alloc_base<_Tp>`, [808](#)
- `__gnu_cxx::per_type_pool_policy<_Tp, _PoolTp, _Thread>`, [818](#)
- `__gnu_cxx::pool<_Thread>`, [818](#)
- `__gnu_cxx::pool<false>`, [818](#)
- `__gnu_cxx::pool<true>`, [819](#)
- `__gnu_cxx::pool_alloc<_Tp>`, [820](#)
- `__gnu_cxx::pool_alloc_base`, [821](#)
- `__gnu_cxx::pool_base`, [822](#)
- `__gnu_cxx::rc_string_base<_CharT, _Traits, _Alloc>`, [823](#)
- `__gnu_cxx::scoped_lock`, [830](#)
- `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`, [835](#)
 - `__versa_string`, [838–841](#)
 - `~__versa_string`, [841](#)
- `append`, [841–844](#)
- `assign`, [844–847](#)
- `at`, [848](#)
- `back`, [848](#), [849](#)
- `begin`, [849](#)
- `c_str`, [849](#)
- `capacity`, [849](#)
- `cbegin`, [849](#)
- `cend`, [850](#)
- `clear`, [850](#)
- `compare`, [850–852](#)
- `copy`, [853](#)
- `crbegin`, [854](#)
- `crend`, [854](#)
- `data`, [854](#)
- `empty`, [854](#)
- `end`, [854](#)
- `erase`, [855](#)
- `find`, [856](#), [857](#)
- `find_first_not_of`, [858](#), [859](#)
- `find_first_of`, [860](#), [861](#)
- `find_last_not_of`, [861–863](#)
- `find_last_of`, [863–865](#)
- `front`, [865](#)

- get_allocator, 865
- insert, 865–870
- length, 871
- max_size, 871
- npos, 887
- operator+=, 871, 872
- operator=, 873, 874
- operator[], 874
- pop_back, 875
- push_back, 875
- rbegin, 875
- rend, 876
- replace, 876–882
- reserve, 883
- resize, 883, 884
- rfind, 884–886
- shrink_to_fit, 886
- size, 886
- substr, 887
- swap, 887
- __gnu_cxx::allocator< _Tp >, 1056
- __gnu_cxx::annotate_base, 1082
- __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1783
- argument_type, 1783
- result_type, 1783
- __gnu_cxx::bitmap_allocator< _Tp >, 1807
- __M_allocate_single_object, 1808
- __M_deallocate_single_object, 1808
- __gnu_cxx::char_traits< _CharT >, 1841
- __gnu_cxx::character< _Value, _Int, _St >, 1849
- __gnu_cxx::condition_base, 1918
- __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >, 1927
- __gnu_cxx::constant_unary_fun< _Result, _Argument >, 1928
- __gnu_cxx::constant_void_fun< _Result >, 1929
- __gnu_cxx::debug_allocator< _Alloc >, 2048
- __gnu_cxx::enc_filebuf< _CharT >, 2120
- __M_buf, 2135
- __M_buf_locale, 2135
- __M_buf_size, 2135
- __M_create_pback, 2123
- __M_destroy_pback, 2123
- __M_ext_buf, 2136
- __M_ext_buf_size, 2136
- __M_ext_next, 2136
- __M_in_beg, 2136
- __M_in_cur, 2136
- __M_in_end, 2136
- __M_mode, 2136
- __M_out_beg, 2136
- __M_out_cur, 2137
- __M_out_end, 2137
- __M_pback, 2137
- __M_pback_cur_save, 2137
- __M_pback_end_save, 2137
- __M_pback_init, 2137
- __M_reading, 2137
- __M_set_buffer, 2123
- close, 2123
- eback, 2123
- egptr, 2123
- epptr, 2124
- gbump, 2124
- getloc, 2124
- gptr, 2124
- imbue, 2125
- in_avail, 2125
- is_open, 2125
- open, 2125
- overflow, 2126
- pbackfail, 2127
- pbase, 2127
- pbump, 2127
- pptr, 2128
- pubimbue, 2128
- pubseekoff, 2128
- pubseekpos, 2129
- pubsetbuf, 2129
- pubsync, 2129
- sbumpc, 2129
- seekoff, 2129
- seekpos, 2130
- setbuf, 2130
- setg, 2130
- setp, 2131
- sgetc, 2131
- sgetn, 2131
- showmanyc, 2132
- snextc, 2132
- sputbackc, 2132
- sputc, 2133
- sputn, 2133
- sungetc, 2133
- sync, 2133
- uflow, 2134
- underflow, 2134
- xsgetrn, 2134
- xspn, 2135
- __gnu_cxx::encoding_char_traits< _CharT >, 2138
- __gnu_cxx::encoding_state, 2139
- __gnu_cxx::equal_to< _Tp >, 2144
- first_argument_type, 2144
- result_type, 2144
- second_argument_type, 2144
- __gnu_cxx::forced_error, 2174
- what, 2174

- __gnu_cxx::free_list, [2198](#)
- _M_clear, [2199](#)
- _M_get, [2199](#)
- _M_insert, [2199](#)
- __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, [2265](#)
- __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, [2267](#)
- __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >, [2268](#)
- __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >, [2270](#)
- __gnu_cxx::limit_condition, [2357](#)
- __gnu_cxx::limit_condition::always_adjustor, [1082](#)
- __gnu_cxx::limit_condition::limit_adjustor, [2357](#)
- __gnu_cxx::limit_condition::never_adjustor, [2565](#)
- __gnu_cxx::malloc_allocator< _Tp >, [2416](#)
- __gnu_cxx::new_allocator< _Tp >, [2566](#)
- __gnu_cxx::pair< _T1, _T2 >, [2719](#)
- explicit, [2722](#)
- first, [2724](#)
- first_type, [2721](#)
- operator=, [2723](#), [2724](#)
- pair, [2721](#)
- second, [2724](#)
- second_type, [2721](#)
- swap, [2724](#)
- __gnu_cxx::project1st< _Arg1, _Arg2 >, [2788](#)
- first_argument_type, [2789](#)
- result_type, [2789](#)
- second_argument_type, [2789](#)
- __gnu_cxx::project2nd< _Arg1, _Arg2 >, [2789](#)
- first_argument_type, [2789](#)
- result_type, [2789](#)
- second_argument_type, [2790](#)
- __gnu_cxx::random_condition, [2798](#)
- __gnu_cxx::random_condition::always_adjustor, [1082](#)
- __gnu_cxx::random_condition::group_adjustor, [2240](#)
- __gnu_cxx::random_condition::never_adjustor, [2565](#)
- __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >, [2815](#)
- __gnu_cxx::recursive_init_error, [2829](#)
- what, [2829](#)
- __gnu_cxx::rope< _CharT, _Alloc >, [2857](#)
- __gnu_cxx::select1st< _Pair >, [2880](#)
- argument_type, [2881](#)
- result_type, [2881](#)
- __gnu_cxx::select2nd< _Pair >, [2881](#)
- argument_type, [2881](#)
- result_type, [2881](#)
- __gnu_cxx::slist< _Tp, _Alloc >, [2933](#)
- __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2944](#)
- _M_buf, [2964](#)
- _M_buf_locale, [2964](#)
- _M_buf_size, [2964](#)
- _M_create_pback, [2948](#)
- _M_destroy_pback, [2948](#)
- _M_ext_buf, [2964](#)
- _M_ext_buf_size, [2964](#)
- _M_ext_next, [2965](#)
- _M_in_beg, [2965](#)
- _M_in_cur, [2965](#)
- _M_in_end, [2965](#)
- _M_mode, [2965](#)
- _M_out_beg, [2965](#)
- _M_out_cur, [2965](#)
- _M_out_end, [2965](#)
- _M_pback, [2965](#)
- _M_pback_cur_save, [2966](#)
- _M_pback_end_save, [2966](#)
- _M_pback_init, [2966](#)
- _M_reading, [2966](#)
- _M_set_buffer, [2948](#)
- ~stdio_filebuf, [2948](#)
- close, [2948](#)
- eback, [2949](#)
- egptr, [2949](#)
- epptr, [2949](#)
- fd, [2950](#)
- file, [2950](#)
- gbump, [2950](#)
- getloc, [2950](#)
- gptr, [2951](#)
- imbue, [2951](#)
- in_avail, [2951](#)
- is_open, [2952](#)
- open, [2952](#), [2953](#)
- overflow, [2954](#)
- pbackfail, [2954](#)
- pbase, [2955](#)
- pbump, [2955](#)
- pptr, [2955](#)
- pubimbue, [2955](#)
- pubseekoff, [2956](#)
- pubseekpos, [2956](#)
- pubsetbuf, [2956](#)
- pubsync, [2957](#)
- sbumpc, [2957](#)
- seekoff, [2957](#)
- seekpos, [2957](#)
- setbuf, [2958](#)
- setg, [2958](#)
- setp, [2959](#)
- sgetc, [2959](#)
- sgetn, [2959](#)
- showmanyc, [2960](#)
- snextc, [2960](#)
- sputbackc, [2960](#)

- sputc, 2961
- sputn, 2961
- stdio_filebuf, 2947
- sungetc, 2961
- sync, 2962
- uflow, 2962
- underflow, 2962
- xsggetn, 2963
- xspn, 2963
- __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2967
 - _M_buf_locale, 2982
 - _M_in_beg, 2982
 - _M_in_cur, 2982
 - _M_in_end, 2982
 - _M_out_beg, 2982
 - _M_out_cur, 2982
 - _M_out_end, 2982
- eback, 2969
- egptr, 2969
- eptr, 2969
- file, 2970
- gbump, 2970
- getloc, 2970
- gptr, 2970
- imbue, 2971
- in_avail, 2971
- overflow, 2971
- pbackfail, 2972
- pbase, 2972
- pbump, 2973
- pptr, 2973
- pubimbue, 2973
- pubseekoff, 2974
- pubseekpos, 2974
- pubsetbuf, 2974
- pubsync, 2974
- sbumpc, 2975
- seekoff, 2975
- seekpos, 2975
- setbuf, 2975
- setg, 2976
- setp, 2976
- sgetc, 2977
- sgetn, 2977
- showmanyc, 2977
- snextc, 2978
- sputbackc, 2978
- sputc, 2978
- sputn, 2979
- sungetc, 2979
- sync, 2979
- uflow, 2980
- underflow, 2980
- xsggetn, 2980
- xspn, 2981
- __gnu_cxx::subtractive_rng, 2999
 - argument_type, 3000
 - operator(), 3000
 - result_type, 3000
 - subtractive_rng, 3000
- __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3003
 - _M_requested_size, 3005
 - ~temporary_buffer, 3004
 - begin, 3005
 - end, 3005
 - size, 3005
 - temporary_buffer, 3004
- __gnu_cxx::throw_allocator_base< _Tp, _Cond >, 3009
- __gnu_cxx::throw_allocator_limit< _Tp >, 3010
- __gnu_cxx::throw_allocator_random< _Tp >, 3012
- __gnu_cxx::throw_value_base< _Cond >, 3013
- __gnu_cxx::throw_value_limit, 3014
- __gnu_cxx::throw_value_random, 3015
- __gnu_cxx::typelist, 415
 - apply_generator, 416
- __gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3094
 - argument_type, 3095
 - result_type, 3095
- __gnu_debug, 416
 - _Distance_precision, 422
 - _base, 422
 - _check_singular, 422
 - _check_singular_aux, 422
 - _check_string, 422
 - _foreign_iterator_aux2, 423
 - _get_distance, 423
 - _valid_range, 423
 - u16string, 422
 - u32string, 422
- __gnu_debug:: After_nth_from< _Iterator >, 888
- __gnu_debug:: BeforeBeginHelper< _Sequence >, 891
- __gnu_debug:: Equal_to< _Type >, 903
- __gnu_debug:: Not_equal_to< _Type >, 969
- __gnu_debug:: Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >, 986
- __gnu_debug:: Safe_forward_list< _SafeSequence >, 988
 - _M_const_iterators, 989
 - _M_detach_all, 989
 - _M_detach_singular, 989
 - _M_get_mutex, 989
 - _M_invalidate_if, 989
 - _M_iterators, 989
 - _M_revalidate_singular, 989
 - _M_swap, 989
 - _M_transfer_from_if, 989

- [_M_version](#), 990
- [__gnu_debug:: Safe_iterator< _Iterator, _Sequence, _Category >](#), 990
- [_M_attach](#), 993
- [_M_attach_single](#), 994
- [_M_attached_to](#), 994
- [_M_before_dereferenceable](#), 994
- [_M_can_compare](#), 994
- [_M_dereferenceable](#), 994
- [_M_detach](#), 995
- [_M_detach_single](#), 995
- [_M_get_mutex](#), 995
- [_M_incrementable](#), 995
- [_M_invalidate](#), 995
- [_M_is_before_begin](#), 995
- [_M_is_begin](#), 995
- [_M_is_beginnest](#), 996
- [_M_is_end](#), 996
- [_M_next](#), 998
- [_M_prior](#), 998
- [_M_reset](#), 996
- [_M_sequence](#), 999
- [_M_singular](#), 996
- [_M_unlink](#), 996
- [_M_value_initialized](#), 996
- [_M_version](#), 999
- [_S_constant](#), 997
- [_Safe_iterator](#), 992, 993
- [base](#), 997
- [operator _Iterator](#), 997
- [operator++](#), 997, 998
- [operator->](#), 998
- [operator=](#), 998
- [operator*](#), 997
- [__gnu_debug:: Safe_iterator_base](#), 999
 - [_M_attach](#), 1002
 - [_M_attach_single](#), 1002
 - [_M_attached_to](#), 1002
 - [_M_can_compare](#), 1002
 - [_M_detach](#), 1002
 - [_M_detach_single](#), 1002
 - [_M_get_mutex](#), 1003
 - [_M_invalidate](#), 1003
 - [_M_next](#), 1003
 - [_M_prior](#), 1003
 - [_M_reset](#), 1003
 - [_M_sequence](#), 1004
 - [_M_singular](#), 1003
 - [_M_unlink](#), 1003
 - [_M_version](#), 1004
 - [_Safe_iterator_base](#), 1001
- [__gnu_debug:: Safe_local_iterator< _Iterator, _UContainer >](#), 1004
 - [_M_attach](#), 1008
 - [_M_attach_single](#), 1008, 1009
 - [_M_attached_to](#), 1009
 - [_M_can_compare](#), 1009
 - [_M_dereferenceable](#), 1009
 - [_M_detach](#), 1009
 - [_M_detach_single](#), 1009
 - [_M_get_mutex](#), 1009
 - [_M_in_same_bucket](#), 1009
 - [_M_incrementable](#), 1010
 - [_M_invalidate](#), 1010
 - [_M_is_begin](#), 1010
 - [_M_is_end](#), 1010
 - [_M_next](#), 1013
 - [_M_prior](#), 1013
 - [_M_reset](#), 1010
 - [_M_sequence](#), 1013
 - [_M_singular](#), 1010
 - [_M_unlink](#), 1010
 - [_M_value_initialized](#), 1011
 - [_M_version](#), 1013
 - [_S_constant](#), 1011
 - [_Safe_local_iterator](#), 1007
 - [base](#), 1011
 - [bucket](#), 1011
 - [operator _Iterator](#), 1011
 - [operator++](#), 1012
 - [operator->](#), 1012
 - [operator=](#), 1012
 - [operator*](#), 1011
- [__gnu_debug:: Safe_local_iterator_base](#), 1013
 - [_M_attach](#), 1016
 - [_M_attach_single](#), 1016
 - [_M_attached_to](#), 1016
 - [_M_can_compare](#), 1016
 - [_M_detach](#), 1017
 - [_M_detach_single](#), 1017
 - [_M_get_mutex](#), 1017
 - [_M_invalidate](#), 1017
 - [_M_next](#), 1018
 - [_M_prior](#), 1018
 - [_M_reset](#), 1017
 - [_M_sequence](#), 1018
 - [_M_singular](#), 1017
 - [_M_unlink](#), 1017
 - [_M_version](#), 1018
 - [_Safe_local_iterator_base](#), 1015
- [__gnu_debug:: Safe_node_sequence< _Sequence >](#), 1018
 - [_M_const_iterators](#), 1021
 - [_M_detach_all](#), 1020
 - [_M_detach_singular](#), 1020
 - [_M_get_mutex](#), 1020
 - [_M_invalidate_if](#), 1020
 - [_M_iterators](#), 1021

- [_M_revalidate_singular](#), 1020
- [_M_swap](#), 1020
- [_M_transfer_from_if](#), 1020
- [_M_version](#), 1021
- [__gnu_debug:: Safe_sequence< _Sequence >](#), 1021
 - [_M_const_iterators](#), 1024
 - [_M_detach_all](#), 1023
 - [_M_detach_singular](#), 1023
 - [_M_get_mutex](#), 1023
 - [_M_invalidate_all](#), 1023
 - [_M_invalidate_if](#), 1023
 - [_M_iterators](#), 1024
 - [_M_revalidate_singular](#), 1023
 - [_M_swap](#), 1023
 - [_M_transfer_from_if](#), 1024
 - [_M_version](#), 1024
- [__gnu_debug:: Safe_sequence_base](#), 1024
 - [_M_const_iterators](#), 1027
 - [_M_detach_all](#), 1026
 - [_M_detach_singular](#), 1026
 - [_M_get_mutex](#), 1026
 - [_M_invalidate_all](#), 1027
 - [_M_iterators](#), 1027
 - [_M_revalidate_singular](#), 1027
 - [_M_swap](#), 1027
 - [_M_version](#), 1027
 - [~_Safe_sequence_base](#), 1026
- [__gnu_debug:: Safe_unordered_container< _Container >](#), 1027
 - [_M_const_iterators](#), 1030
 - [_M_const_local_iterators](#), 1030
 - [_M_detach_all](#), 1029
 - [_M_detach_singular](#), 1029
 - [_M_get_mutex](#), 1029
 - [_M_invalidate_all](#), 1029
 - [_M_iterators](#), 1030
 - [_M_local_iterators](#), 1030
 - [_M_revalidate_singular](#), 1029
 - [_M_swap](#), 1029
 - [_M_version](#), 1030
- [__gnu_debug:: Safe_unordered_container_base](#), 1030
 - [_M_const_iterators](#), 1033
 - [_M_const_local_iterators](#), 1033
 - [_M_detach_all](#), 1032
 - [_M_detach_singular](#), 1032
 - [_M_get_mutex](#), 1032
 - [_M_invalidate_all](#), 1033
 - [_M_iterators](#), 1033
 - [_M_local_iterators](#), 1033
 - [_M_revalidate_singular](#), 1033
 - [_M_swap](#), 1033
 - [_M_version](#), 1034
 - [~_Safe_unordered_container_base](#), 1032
- [__gnu_debug:: Safe_vector< _SafeSequence, _BaseSequence >](#), 1034
- [__gnu_debug:: Sequence_traits< _Sequence >](#), 1037
- [__gnu_debug:: basic_string< _CharT, _Traits, _Allocator >](#), 1512
 - [__resize_and_overwrite](#), 1519
 - [append](#), 1519–1521
 - [assign](#), 1522, 1523
 - [at](#), 1524–1526
 - [back](#), 1527
 - [capacity](#), 1527
 - [compare](#), 1527–1533
 - [copy](#), 1534
 - [data](#), 1534
 - [empty](#), 1534
 - [erase](#), 1534–1536
 - [find](#), 1536–1541
 - [find_first_not_of](#), 1541–1545
 - [find_first_of](#), 1546–1550
 - [find_last_not_of](#), 1551–1555
 - [find_last_of](#), 1556–1560
 - [front](#), 1560
 - [get_allocator](#), 1561
 - [insert](#), 1561–1567
 - [length](#), 1568
 - [max_size](#), 1568
 - [npos](#), 1589
 - [operator+=](#), 1568
 - [operator\[\]](#), 1568, 1569
 - [replace](#), 1569–1579
 - [reserve](#), 1580, 1581
 - [resize](#), 1582
 - [rfind](#), 1582–1586
 - [size](#), 1587
 - [substr](#), 1587
 - [subview](#), 1588
 - [swap](#), 1588
- [__gnu_debug:: type_info](#), 3089
 - [~type_info](#), 3090
 - [before](#), 3090
 - [name](#), 3090
- [__gnu_internal](#), 424
- [__gnu_parallel](#), 424
 - [_AlgorithmStrategy](#), 432
 - [_BinIndex](#), 432
 - [_CASable](#), 432
 - [_CASable_bits](#), 473
 - [_CASable_mask](#), 473
 - [_FindAlgorithm](#), 432
 - [_MultiwayMergeAlgorithm](#), 432
 - [_Parallelism](#), 432
 - [_PartialSumAlgorithm](#), 432
 - [_SequenceIndex](#), 432
 - [_SortAlgorithm](#), 433

- `__SplittingAlgorithm`, 433
- `__ThreadIndex`, 432
- `__calc_borders`, 433
- `__compare_and_swap`, 433
- `__decode2`, 433
- `__determine_samples`, 434
- `__encode2`, 434
- `__equally_split`, 435
- `__equally_split_point`, 435
- `__fetch_and_add`, 436
- `__find_template`, 436–438
- `__for_each_template_random_access`, 438
- `__for_each_template_random_access_ed`, 439
- `__for_each_template_random_access_omp_loop`, 440
- `__for_each_template_random_access_omp_loop_static`, 441
- `__for_each_template_random_access_workstealing`, 442
- `__is_sorted`, 442
- `__median_of_three_iterators`, 443
- `__merge_advance`, 443
- `__merge_advance_movc`, 444
- `__merge_advance_usual`, 445
- `__parallel_merge_advance`, 445, 446
- `__parallel_nth_element`, 446
- `__parallel_partial_sort`, 447
- `__parallel_partial_sum`, 447
- `__parallel_partial_sum_basecase`, 448
- `__parallel_partial_sum_linear`, 448
- `__parallel_partition`, 449
- `__parallel_random_shuffle`, 449
- `__parallel_random_shuffle_drs`, 449
- `__parallel_random_shuffle_drs_pu`, 450
- `__parallel_sort`, 450–455
- `__parallel_sort_qs`, 455
- `__parallel_sort_qs_conquer`, 456
- `__parallel_sort_qs_divide`, 456
- `__parallel_sort_qsb`, 457
- `__parallel_unique_copy`, 457, 458
- `__qsb_conquer`, 458
- `__qsb_divide`, 459
- `__qsb_local_sort_with_helping`, 459
- `__random_number_pow2`, 459
- `__rd_log2`, 460
- `__round_up_to_pow2`, 460
- `__search_template`, 460
- `__sequential_multiway_merge`, 461
- `__sequential_random_shuffle`, 461
- `__shrink`, 462
- `__shrink_and_double`, 462
- `__yield`, 463
- `list_partition`, 463
- `max`, 463
- `min`, 464
- `multiseq_partition`, 464
- `multiseq_selection`, 464
- `multiway_merge`, 465
- `multiway_merge_3_variant`, 466
- `multiway_merge_4_variant`, 467
- `multiway_merge_exact_splitting`, 468
- `multiway_merge_loser_tree`, 468
- `multiway_merge_loser_tree_sentinel`, 468
- `multiway_merge_loser_tree_unguarded`, 469
- `multiway_merge_sampling_splitting`, 470
- `multiway_merge_sentinels`, 470
- `parallel_balanced`, 432
- `parallel_multiway_merge`, 472
- `parallel_omp_loop`, 432
- `parallel_omp_loop_static`, 432
- `parallel_sort_mwms`, 472
- `parallel_sort_mwms_pu`, 473
- `parallel_taskqueue`, 432
- `parallel_unbalanced`, 432
- `sequential`, 432
- `__gnu_parallel::DRSSorterPU<_RAIter, _RandomNumberGenerator>`, 902
 - `_M_bins_begin`, 903
 - `_M_num_threads`, 903
 - `_M_sd`, 903
 - `_M_seed`, 903
 - `_bins_end`, 902
- `__gnu_parallel::DRandomShufflingGlobalData<_RAIter>`, 900
 - `_DRandomShufflingGlobalData`, 901
 - `_M_bin_proc`, 901
 - `_M_dist`, 901
 - `_M_num_bins`, 901
 - `_M_num_bits`, 901
 - `_M_source`, 902
 - `_M_starts`, 902
 - `_M_temporaries`, 902
- `__gnu_parallel::_DummyReduct`, 903
- `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`, 904
 - `first_argument_type`, 905
 - `result_type`, 905
 - `second_argument_type`, 905
- `__gnu_parallel::_EqualTo<_T1, _T2>`, 905
 - `first_argument_type`, 906
 - `result_type`, 906
 - `second_argument_type`, 906
- `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>`, 915
 - `_GuardedIterator`, 915
 - `operator _RAIter`, 916
 - `operator<`, 916
 - `operator<=`, 916

- operator++, 916
- operator*, 916
- __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 922
- explicit, 925–927
- first, 928
- first_type, 925
- operator<=, 927
- operator==, 927
- second, 928
- second_type, 925
- swap, 927, 928
- __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >, 929
- __gnu_parallel::_Job< _DifferenceTp >, 929
 - _M_first, 930
 - _M_last, 930
 - _M_load, 930
- __gnu_parallel::_Less< _T1, _T2 >, 932
 - first_argument_type, 932
 - result_type, 932
 - second_argument_type, 932
- __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 933
 - first_argument_type, 933
 - result_type, 934
 - second_argument_type, 934
- __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 934
 - first_argument_type, 935
 - result_type, 935
 - second_argument_type, 935
- __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 943
 - _M_log_k, 944
 - __delete_min_insert, 943
 - __get_min_source, 944
 - __insert_start, 944
- __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 944
 - _M_log_k, 946
 - __delete_min_insert, 945
 - __get_min_source, 946
 - __init_winner, 946
 - __insert_start, 946
- __gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 947
 - _LoserTreeBase, 948
 - _M_comp, 949
 - _M_first_insert, 949
 - _M_log_k, 949
 - _M_losers, 949
 - __get_min_source, 948
 - __insert_start, 948
 - ~_LoserTreeBase, 948
- __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 942
 - _M_key, 942
 - _M_source, 942
 - _M_sup, 942
- __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >, 949
- __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 950
- __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >, 951
- __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser, 942
- __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >, 952
- __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >, 953
- __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >, 954
- __gnu_parallel::_LoserTreeTraits< _Tp >, 955
 - _M_use_pointer, 956
- __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >, 956
- __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 957
- __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >, 958
- __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 960
 - first_argument_type, 961
 - result_type, 961
 - second_argument_type, 961
- __gnu_parallel::_Nothing, 970
- operator(), 971
- __gnu_parallel::_PMWMSSortingData< _RAIter >, 973
 - _M_num_threads, 974
 - _M_offsets, 974
 - _M_pieces, 974
 - _M_samples, 974
 - _M_source, 974
 - _M_starts, 974
 - _M_temporary, 974
- __gnu_parallel::_Piece< _DifferenceTp >, 971
 - _M_begin, 971
 - _M_end, 971
- __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 972
 - first_argument_type, 973
 - result_type, 973
 - second_argument_type, 973
- __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 977
 - _PseudoSequence, 977
 - begin, 978
 - end, 978

- `__gnu_parallel::PseudoSequencerIterator< _Tp, _DifferenceTp >`, 978
- `__gnu_parallel::QSBThreadLocal< _RAIter >`, 979
 - `_M_elements_leftover`, 980
 - `_M_global`, 980
 - `_M_initial`, 980
 - `_M_leftover_parts`, 980
 - `_M_num_threads`, 980
 - `_Piece`, 979
 - `_QSBThreadLocal`, 979
- `__gnu_parallel::RandomNumber`, 981
 - `_RandomNumber`, 981
 - `_genrand_bits`, 982
 - `operator()`, 982
- `__gnu_parallel::RestrictedBoundedConcurrentQueue< _Tp >`, 984
 - `_RestrictedBoundedConcurrentQueue`, 985
 - `~_RestrictedBoundedConcurrentQueue`, 985
 - `pop_back`, 985
 - `pop_front`, 985
 - `push_front`, 986
- `__gnu_parallel::SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`, 1035
- `__gnu_parallel::SamplingSorter< false, _RAIter, _StrictWeakOrdering >`, 1035
- `__gnu_parallel::Settings`, 1037
 - `accumulate_minimal_n`, 1039
 - `adjacent_difference_minimal_n`, 1039
 - `cache_line_size`, 1039
 - `count_minimal_n`, 1039
 - `fill_minimal_n`, 1040
 - `find_increasing_factor`, 1040
 - `find_initial_block_size`, 1040
 - `find_maximum_block_size`, 1040
 - `find_scale_factor`, 1040
 - `find_sequential_search_size`, 1040
 - `for_each_minimal_n`, 1040
 - `generate_minimal_n`, 1040
 - `get`, 1039
 - `L1_cache_size`, 1040
 - `L2_cache_size`, 1041
 - `max_element_minimal_n`, 1041
 - `merge_minimal_n`, 1041
 - `merge_oversampling`, 1041
 - `min_element_minimal_n`, 1041
 - `multiway_merge_minimal_k`, 1041
 - `multiway_merge_minimal_n`, 1041
 - `multiway_merge_oversampling`, 1041
 - `nth_element_minimal_n`, 1041
 - `partial_sort_minimal_n`, 1042
 - `partial_sum_dilation`, 1042
 - `partial_sum_minimal_n`, 1042
 - `partition_chunk_share`, 1042
 - `partition_chunk_size`, 1042
 - `partition_minimal_n`, 1042
 - `qsb_steals`, 1042
 - `random_shuffle_minimal_n`, 1042
 - `replace_minimal_n`, 1042
 - `search_minimal_n`, 1043
 - `set`, 1039
 - `set_difference_minimal_n`, 1043
 - `set_intersection_minimal_n`, 1043
 - `set_symmetric_difference_minimal_n`, 1043
 - `set_union_minimal_n`, 1043
 - `sort_minimal_n`, 1043
 - `sort_mwms_oversampling`, 1043
 - `sort_qs_num_samples_preset`, 1043
 - `sort_qsb_base_case_maximal_n`, 1043
 - `TLB_size`, 1044
 - `transform_minimal_n`, 1044
 - `unique_copy_minimal_n`, 1044
- `__gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >`, 1045
- `__gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`, 1045
- `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`, 1045
- `__gnu_parallel::accumulate_binop_reduct< _BinOp >`, 752
- `__gnu_parallel::accumulate_selector< _It >`, 752
 - `_M_finish_iterator`, 754
 - `operator()`, 753
- `__gnu_parallel::adjacent_difference_selector< _It >`, 754
 - `_M_finish_iterator`, 755
- `__gnu_parallel::adjacent_find_selector`, 755
 - `_M_sequential_algorithm`, 756
 - `operator()`, 756
- `__gnu_parallel::binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 765
 - `argument_type`, 766
 - `result_type`, 766
- `__gnu_parallel::binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 766
 - `argument_type`, 767
 - `result_type`, 767
- `__gnu_parallel::count_if_selector< _It, _Diff >`, 771
 - `_M_finish_iterator`, 773
 - `operator()`, 772
- `__gnu_parallel::count_selector< _It, _Diff >`, 773
 - `_M_finish_iterator`, 774
 - `operator()`, 774
- `__gnu_parallel::fill_selector< _It >`, 790
 - `_M_finish_iterator`, 791
 - `operator()`, 791

[__gnu_parallel::__find_first_of_selector< _FIterator >, 791](#)
[_M_sequential_algorithm, 792](#)
[operator\(\), 792](#)
[__gnu_parallel::__find_if_selector, 793](#)
[_M_sequential_algorithm, 793](#)
[operator\(\), 794](#)
[__gnu_parallel::__for_each_selector< _It >, 794](#)
[_M_finish_iterator, 796](#)
[operator\(\), 795](#)
[__gnu_parallel::__generate_selector< _It >, 796](#)
[_M_finish_iterator, 797](#)
[operator\(\), 797](#)
[__gnu_parallel::__generic_find_selector, 797](#)
[__gnu_parallel::__generic_for_each_selector< _It >, 798](#)
[_M_finish_iterator, 800](#)
[__gnu_parallel::__identity_selector< _It >, 800](#)
[_M_finish_iterator, 801](#)
[operator\(\), 801](#)
[__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 801](#)
[_M_finish_iterator, 803](#)
[__begin1_iterator, 803](#)
[__begin2_iterator, 803](#)
[__inner_product_selector, 802](#)
[operator\(\), 803](#)
[__gnu_parallel::__max_element_reduct< _Compare, _It >, 804](#)
[__gnu_parallel::__min_element_reduct< _Compare, _It >, 804](#)
[__gnu_parallel::__mismatch_selector, 806](#)
[_M_sequential_algorithm, 806](#)
[operator\(\), 807](#)
[__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< _sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 809](#)
[__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 810](#)
[__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< _sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 810](#)
[__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 811](#)
[__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< _sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 811](#)
[__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 811](#)
[__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 827](#)
[_M_finish_iterator, 828](#)
[__new_val, 828](#)
[__replace_if_selector, 828](#)
[operator\(\), 828](#)
[__gnu_parallel::__replace_selector< _It, _Tp >, 829](#)
[_M_finish_iterator, 830](#)
[__new_val, 830](#)
[__replace_selector, 829](#)
[operator\(\), 830](#)
[__gnu_parallel::__transform1_selector< _It >, 831](#)
[_M_finish_iterator, 832](#)
[operator\(\), 831](#)
[__gnu_parallel::__transform2_selector< _It >, 832](#)
[_M_finish_iterator, 833](#)
[operator\(\), 833](#)
[__gnu_parallel::__unary_negate< _Predicate, argument_type >, 833](#)
[argument_type, 834](#)
[result_type, 834](#)
[__gnu_parallel::balanced_quicksort_tag, 1115](#)
[__get_num_threads, 1116](#)
[set_num_threads, 1116](#)
[__gnu_parallel::balanced_tag, 1117](#)
[__get_num_threads, 1117](#)
[set_num_threads, 1117](#)
[__gnu_parallel::constant_size_blocks_tag, 1927](#)
[__gnu_parallel::default_parallel_tag, 2057](#)
[__get_num_threads, 2058](#)
[set_num_threads, 2058](#)
[__gnu_parallel::equal_split_tag, 2143](#)
[__gnu_parallel::exact_tag, 2152](#)
[__get_num_threads, 2153](#)
[set_num_threads, 2153](#)
[__gnu_parallel::find_tag, 2170](#)
[__gnu_parallel::growing_blocks_tag, 2240](#)
[__gnu_parallel::multiway_mergesort_exact_tag, 2555](#)
[__get_num_threads, 2556](#)
[set_num_threads, 2556](#)
[__gnu_parallel::multiway_mergesort_sampling_tag, 2556](#)
[__get_num_threads, 2557](#)
[__gnu_parallel::multiway_mergesort_tag, 2558](#)
[__get_num_threads, 2558](#)
[__gnu_parallel::omp_loop_static_tag, 2693](#)
[__get_num_threads, 2693](#)
[__gnu_parallel::omp_loop_tag, 2694](#)
[__get_num_threads, 2694](#)
[__gnu_parallel::parallel_tag, 2732](#)
[__get_num_threads, 2734](#)
[parallel_tag, 2734](#)
[set_num_threads, 2734](#)
[__gnu_parallel::quicksort_tag, 2796](#)

- [__get_num_threads, 2797](#)
- [set_num_threads, 2797](#)
- [__gnu_parallel::sampling_tag, 2877](#)
- [__get_num_threads, 2877](#)
- [set_num_threads, 2878](#)
- [__gnu_parallel::sequential_tag, 2885](#)
- [__gnu_parallel::unbalanced_tag, 3098](#)
- [__get_num_threads, 3098](#)
- [set_num_threads, 3098](#)
- [__gnu_pbds, 474](#)
- [__gnu_pbds::associative_tag, 1088](#)
- [__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >, 1118](#)
- [__gnu_pbds::basic_branch_tag, 1119](#)
- [__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >, 1191](#)
- [__gnu_pbds::basic_hash_tag, 1193](#)
- [__gnu_pbds::basic_invalidatation_guarantee, 1234](#)
- [__gnu_pbds::binary_heap_tag, 1795](#)
- [__gnu_pbds::binomial_heap_tag, 1807](#)
- [__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1826](#)
- [cc_hash_max_collision_check_resize_trigger, 1828](#)
- [external_load_access, 1827](#)
- [get_load, 1828](#)
- [is_grow_needed, 1828](#)
- [is_resize_needed, 1828](#)
- [notify_cleared, 1828](#)
- [notify_erase_search_collision, 1828](#)
- [notify_erase_search_end, 1828](#)
- [notify_erase_search_start, 1828](#)
- [notify_erased, 1829](#)
- [notify_externally_resized, 1829](#)
- [notify_find_search_collision, 1829](#)
- [notify_find_search_end, 1829](#)
- [notify_find_search_start, 1829](#)
- [notify_insert_search_collision, 1829](#)
- [notify_insert_search_end, 1829](#)
- [notify_insert_search_start, 1830](#)
- [notify_inserted, 1830](#)
- [notify_resized, 1830](#)
- [set_load, 1830](#)
- [__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >, 1830](#)
- [cc_hash_table, 1832–1835](#)
- [__gnu_pbds::cc_hash_tag, 1835](#)
- [__gnu_pbds::container_error, 1938](#)
- [what, 1939](#)
- [__gnu_pbds::container_tag, 1939](#)
- [__gnu_pbds::container_traits< Cntnr >, 1940](#)
- [erase_can_throw, 1941](#)
- [order_preserving, 1941](#)
- [reverse_iteration, 1941](#)
- [split_join_can_throw, 1941](#)
- [__gnu_pbds::container_traits_base< _Tag >, 1941](#)
- [__gnu_pbds::container_traits_base< binary_heap_tag >, 1941](#)
- [__gnu_pbds::container_traits_base< binomial_heap_tag >, 1942](#)
- [__gnu_pbds::container_traits_base< cc_hash_tag >, 1942](#)
- [__gnu_pbds::container_traits_base< gp_hash_tag >, 1942](#)
- [__gnu_pbds::container_traits_base< list_update_tag >, 1943](#)
- [__gnu_pbds::container_traits_base< ov_tree_tag >, 1943](#)
- [__gnu_pbds::container_traits_base< pairing_heap_tag >, 1943](#)
- [__gnu_pbds::container_traits_base< pat_trie_tag >, 1943](#)
- [__gnu_pbds::container_traits_base< rb_tree_tag >, 1944](#)
- [__gnu_pbds::container_traits_base< rc_binomial_heap_tag >, 1944](#)
- [__gnu_pbds::container_traits_base< splay_tree_tag >, 1944](#)
- [__gnu_pbds::container_traits_base< thin_heap_tag >, 1945](#)
- [__gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >, 1772](#)
- [__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1774](#)
- [const_reference, 1775](#)
- [difference_type, 1775](#)
- [get_l_child, 1775](#)
- [get_metadata, 1776](#)
- [get_r_child, 1776](#)
- [iterator_category, 1775](#)
- [metadata_const_reference, 1775](#)
- [metadata_type, 1775](#)
- [operator!=, 1776](#)
- [operator==, 1776](#)
- [operator*, 1776](#)
- [reference, 1775](#)
- [value_type, 1775](#)
- [__gnu_pbds::detail::bin_search_tree_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >, 1776](#)
- [__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1778](#)
- [const_reference, 1779](#)
- [difference_type, 1779](#)
- [get_l_child, 1780](#)

- get_metadata, 1780
- get_r_child, 1780
- iterator_category, 1779
- metadata_const_reference, 1779
- metadata_type, 1780
- operator!=, 1780
- operator==, 1781
- operator*, 1781
- reference, 1780
- value_type, 1780
- __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, 1781
- node_const_iterator, 1782
- __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, 1782
- node_const_iterator, 1782
- __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >, 1786
- __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1788
- binary_heap_const_iterator_, 1790
- const_pointer, 1789
- const_reference, 1789
- difference_type, 1789
- iterator_category, 1789
- operator!=, 1790
- operator->, 1791
- operator==, 1791
- operator*, 1791
- pointer, 1790
- reference, 1790
- value_type, 1790
- __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1791
- binary_heap_point_const_iterator_, 1794
- const_pointer, 1793
- const_reference, 1793
- difference_type, 1793
- iterator_category, 1793
- operator!=, 1794
- operator->, 1794
- operator==, 1794
- operator*, 1794
- pointer, 1793
- reference, 1793
- value_type, 1793
- __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >, 1802
- __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >, 1804
- __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >, 1823
- __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >, 1822
- __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1836
- empty, 1839
- get_comb_hash_fn, 1839
- get_eq_fn, 1840
- get_hash_fn, 1840
- get_resize_policy, 1840
- __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >, 1917
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >, 1929
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >, 1932
- type, 1932
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >, 1932
- type, 1933
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >, 1933
- type, 1933
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >, 1933
- type, 1934
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI >, 1934
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >, 1934
- type, 1934
- __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >, 1935
- type, 1935
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >, 1935
- type, 1935
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >, 1936
- type, 1936
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >, 1936
- type, 1936
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >, 1937
- type, 1937
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >, 1937

- type, [1937](#)
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >, [1938](#)
- __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >, [1938](#)
- type, [1938](#)
- __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >, [1929](#)
- type, [1930](#)
- __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, [1930](#)
- type, [1930](#)
- __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, [1930](#)
- type, [1931](#)
- __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >, [1931](#)
- type, [1931](#)
- __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >, [1931](#)
- type, [1932](#)
- __gnu_pbds::detail::default_comb_hash_fn, [2054](#)
- type, [2054](#)
- __gnu_pbds::detail::default_eq_fn< Key >, [2056](#)
- type, [2056](#)
- __gnu_pbds::detail::default_hash_fn< Key >, [2057](#)
- type, [2057](#)
- __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >, [2058](#)
- type, [2058](#)
- __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >, [2059](#)
- type, [2059](#)
- __gnu_pbds::detail::default_trie_access_traits< Key >, [2059](#)
- __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >, [2060](#)
- type, [2060](#)
- __gnu_pbds::detail::default_update_policy, [2060](#)
- type, [2060](#)
- __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >, [2102](#)
- __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >, [2141](#)
- __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >, [2141](#)
- type, [2142](#)
- __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >, [2142](#)
- __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >, [2142](#)
- __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >, [2142](#)
- __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >, [2143](#)
- __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, [2229](#)
- empty, [2232](#)
- get_comb_probe_fn, [2232](#)
- get_eq_fn, [2233](#)
- get_hash_fn, [2233](#)
- get_probe_fn, [2233](#)
- get_resize_policy, [2233](#), [2234](#)
- __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >, [2259](#)
- __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >, [2259](#)
- __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >, [2260](#)
- __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >, [2264](#)
- __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >, [2264](#)
- __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >, [2342](#)
- __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, [2344](#)
- const_pointer, [2345](#)
- const_reference, [2345](#)
- difference_type, [2345](#)
- iterator_category, [2345](#)
- left_child_next_sibling_heap_const_iterator_, [2345](#), [2346](#)
- operator!=, [2346](#)
- operator->, [2346](#)
- operator==, [2346](#)
- operator*, [2346](#)
- pointer, [2345](#)
- reference, [2345](#)
- value_type, [2345](#)
- __gnu_pbds::detail::left_child_next_sibling_heap_node< _Value, _Metadata, _Alloc >, [2347](#)
- __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, [2347](#)
- const_pointer, [2349](#)
- const_reference, [2349](#)

- difference_type, [2349](#)
- iterator_category, [2349](#)
- left_child_next_sibling_heap_node_point_const_iterator, [2350](#)
- operator!=, [2350](#)
- operator->, [2350](#)
- operator==, [2350](#)
- operator*, [2350](#)
- pointer, [2349](#)
- reference, [2349](#)
- value_type, [2349](#)
- __gnu_pbds::detail::lu_counter_metadata< Size_Type >, [2409](#)
- __gnu_pbds::detail::lu_counter_policy_base< Size_Type >, [2411](#)
- __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >, [2412](#)
- __gnu_pbds::detail::mask_based_range_hashing< Size_Type >, [2444](#)
- __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >, [2454](#)
- __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >, [2456](#)
- __gnu_pbds::detail::mod_based_range_hashing< Size_Type >, [2475](#)
- __gnu_pbds::detail::no_throw_copies< Key, Mapped >, [2567](#)
- __gnu_pbds::detail::no_throw_copies< Key, null_type >, [2567](#)
- __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2706](#)
- node_begin, [2708](#)
- node_end, [2708](#)
- __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >, [1918](#)
- __gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >, [2709](#)
- get_l_child, [2710](#)
- get_r_child, [2710](#)
- __gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >, [2711](#)
- get_l_child, [2712](#)
- get_r_child, [2712](#)
- operator*, [2712](#)
- __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >, [2730](#)
- __gnu_pbds::detail::pat_trie_base, [2745](#)
- node_type, [2746](#)
- __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, [894](#)
- __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >, [917](#)
- __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >, [918](#)
- __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator, [1920](#)
- __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator, [2336](#)
- __gnu_pbds::detail::pat_trie_base::_lter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, [920](#)
- __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >, [930](#)
- __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >, [959](#)
- __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >, [959](#)
- __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >, [962](#)
- __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, [964](#)
- get_child, [965](#)
- get_metadata, [965](#)
- metadata_const_reference, [965](#)
- metadata_type, [965](#)
- num_children, [965](#)
- operator!=, [965](#)
- operator==, [966](#)
- operator*, [966](#)
- valid_prefix, [966](#)
- __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, [966](#)
- get_child, [968](#)
- get_metadata, [968](#)
- metadata_const_reference, [968](#)
- metadata_type, [968](#)
- num_children, [968](#)
- operator!=, [968](#)
- operator==, [969](#)
- operator*, [969](#)
- valid_prefix, [969](#)
- __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, [2746](#)
- node_begin, [2748](#)
- node_end, [2749](#)
- node_type, [2748](#)
- __gnu_pbds::detail::probe_fn_base< _Alloc >, [2788](#)
- __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >, [2802](#)
- __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >, [2802](#)
- __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >, [2803](#)
- __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >, [2803](#)
- __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >, [2803](#)

- loc, Comb_Hash_Fn, true >, [2804](#)
- __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >, [2805](#)
- __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >, [2805](#)
- __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >, [2806](#)
- __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >, [2807](#)
- __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2818](#)
- node_begin, [2821](#)
- node_end, [2821](#)
- __gnu_pbds::detail::rb_tree_node< Value_Type, Meta-data, _Alloc >, [2822](#)
- __gnu_pbds::detail::rc< _Node, _Alloc >, [2823](#)
- __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >, [2824](#)
- __gnu_pbds::detail::rebind_traits< _Alloc, T >, [2827](#)
- __gnu_pbds::detail::resize_policy< _Tp >, [2851](#)
- __gnu_pbds::detail::select_value_type< Key, Mapped >, [2882](#)
- __gnu_pbds::detail::select_value_type< Key, null_type >, [2882](#)
- __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2936](#)
- node_begin, [2939](#)
- node_end, [2939](#)
- __gnu_pbds::detail::splay_tree_node< Value_Type, Metadata, _Alloc >, [2939](#)
- __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >, [2983](#)
- __gnu_pbds::detail::stored_data< _Tv, _Th, false >, [2984](#)
- __gnu_pbds::detail::stored_hash< _Th >, [2985](#)
- __gnu_pbds::detail::stored_value< _Tv >, [2985](#)
- __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >, [3000](#)
- __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >, [3005](#)
- __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >, [3053](#)
- __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >, [3053](#)
- __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >, [3053](#)
- __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, [3054](#)
- __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >, [3056](#)
- __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [3057](#)
- node_const_iterator, [3057](#)
- __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [3057](#)
- node_const_iterator, [3059](#)
- __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [3059](#)
- node_const_iterator, [3060](#)
- __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [3060](#)
- node_const_iterator, [3061](#)
- __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [3061](#)
- node_const_iterator, [3062](#)
- __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [3062](#)
- node_const_iterator, [3063](#)
- __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >, [3066](#)
- __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >, [3066](#)
- __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >, [3066](#)
- __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, [3067](#)
- __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >, [3070](#)
- end, [3071](#)
- __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >, [3077](#)
- __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3078](#)
- node_const_iterator, [3078](#)
- node_update, [3078](#)
- synth_access_traits, [3078](#)
- __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3079](#)
- node_const_iterator, [3079](#)
- node_update, [3079](#)
- synth_access_traits, [3080](#)
- __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >, [3092](#)
- __gnu_pbds::direct_mask_range_hashing< Size_Type >, [2086](#)
- operator(), [2087](#)
- __gnu_pbds::direct_mod_range_hashing< Size_Type >, [2087](#)
- operator(), [2088](#)
- __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Re-size_Policy, Store_Hash, _Alloc >, [2223](#)
- gp_hash_table, [2224–2228](#)
- __gnu_pbds::gp_hash_tag, [2228](#)
- __gnu_pbds::hash_exponential_size_policy< Size_Type

- >, 2260
- hash_exponential_size_policy, 2261
- __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2261
- external_load_access, 2263
- get_loads, 2263
- hash_load_check_resize_trigger, 2263
- notify_cleared, 2263
- notify_inserted, 2263
- notify_resized, 2263
- set_loads, 2264
- __gnu_pbds::hash_prime_size_policy, 2270
- hash_prime_size_policy, 2270
- size_type, 2270
- __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2272
- external_load_access, 2273
- get_actual_size, 2274
- get_loads, 2274
- get_new_size, 2274
- get_size_policy, 2275
- get_trigger_policy, 2275
- hash_standard_resize_policy, 2274
- resize, 2275
- set_loads, 2275
- __gnu_pbds::insert_error, 2288
- what, 2289
- __gnu_pbds::join_error, 2341
- what, 2342
- __gnu_pbds::linear_probe_fn< Size_Type >, 2362
- operator(), 2363
- __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, 2387
- list_update, 2388
- __gnu_pbds::list_update_tag, 2388
- __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2409
- max_count, 2411
- metadata_reference, 2410
- metadata_type, 2410
- operator(), 2411
- __gnu_pbds::lu_move_to_front_policy< _Alloc >, 2415
- metadata_reference, 2415
- metadata_type, 2415
- operator(), 2415
- __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >, 2574
- __gnu_pbds::null_type, 2575
- __gnu_pbds::ov_tree_tag, 2712
- __gnu_pbds::pairing_heap_tag, 2732
- __gnu_pbds::pat_trie_tag, 2749
- __gnu_pbds::point_invalidation_guarantee, 2769
- __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, 2781
- priority_queue, 2782
- __gnu_pbds::priority_queue_tag, 2788
- __gnu_pbds::quadratic_probe_fn< Size_Type >, 2793
- operator(), 2793
- __gnu_pbds::range_invalidation_guarantee, 2801
- __gnu_pbds::rb_tree_tag, 2822
- __gnu_pbds::rc_binomial_heap_tag, 2826
- __gnu_pbds::resize_error, 2850
- what, 2851
- __gnu_pbds::sample_probe_fn, 2863
- operator(), 2864
- sample_probe_fn, 2864
- swap, 2864
- __gnu_pbds::sample_range_hashing, 2864
- notify_resized, 2865
- operator(), 2865
- sample_range_hashing, 2865
- size_type, 2865
- swap, 2865
- __gnu_pbds::sample_ranged_hash_fn, 2865
- notify_resized, 2866
- operator(), 2866
- sample_ranged_hash_fn, 2866
- swap, 2866
- __gnu_pbds::sample_ranged_probe_fn, 2866
- __gnu_pbds::sample_resize_policy, 2867
- get_new_size, 2868
- is_resize_needed, 2868
- notify_cleared, 2868
- notify_erase_search_collision, 2868
- notify_erase_search_end, 2868
- notify_erase_search_start, 2868
- notify_erased, 2868
- notify_find_search_collision, 2868
- notify_find_search_end, 2869
- notify_find_search_start, 2869
- notify_insert_search_collision, 2869
- notify_insert_search_end, 2869
- notify_insert_search_start, 2869
- notify_inserted, 2869
- notify_resized, 2869
- sample_range_hashing, 2869
- sample_resize_policy, 2868
- size_type, 2868
- swap, 2869
- __gnu_pbds::sample_resize_trigger, 2870
- is_grow_needed, 2871
- is_resize_needed, 2871
- notify_cleared, 2871
- notify_erase_search_collision, 2871
- notify_erase_search_end, 2871
- notify_erase_search_start, 2871

- notify_erased, 2871
- notify_externally_resized, 2871
- notify_find_search_collision, 2871
- notify_find_search_end, 2871
- notify_find_search_start, 2871
- notify_insert_search_collision, 2872
- notify_insert_search_end, 2872
- notify_insert_search_start, 2872
- notify_inserted, 2872
- notify_resized, 2872
- sample_range_hashing, 2872
- sample_resize_trigger, 2870
- size_type, 2870
- swap, 2872
- __gnu_pbds::sample_size_policy, 2872
 - get_nearest_larger_size, 2873
 - get_nearest_smaller_size, 2873
 - sample_range_hashing, 2873
 - sample_size_policy, 2873
 - size_type, 2873
 - swap, 2873
- __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >, 2874
- __gnu_pbds::sample_trie_access_traits, 2874
 - begin, 2874
 - e_pos, 2874
 - e_type, 2874
 - end, 2875
- __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 2875
 - operator(), 2875
 - sample_trie_node_update, 2875
- __gnu_pbds::sample_update_policy, 2876
 - metadata_type, 2876
 - operator(), 2876
 - sample_update_policy, 2876
 - swap, 2877
- __gnu_pbds::sequence_tag, 2885
- __gnu_pbds::splay_tree_tag, 2940
- __gnu_pbds::string_tag, 2986
- __gnu_pbds::thin_heap_tag, 3007
- __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3050
 - cmp_fn, 3052
 - tree, 3052
- __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >, 3054
 - find_by_order, 3055
 - operator(), 3055
 - order_of_key, 3055
- __gnu_pbds::tree_tag, 3056
- __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3063
 - access_traits, 3065
 - trie, 3065
- __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3067
 - find_by_order, 3069
 - operator(), 3069
 - order_of_key, 3069
 - order_of_prefix, 3069
- __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3071
 - a_const_iterator, 3073
 - access_traits, 3073
 - allocator_type, 3073
 - operator(), 3073
 - prefix_range, 3073, 3074
 - size_type, 3073
- __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3074
 - begin, 3075
 - const_iterator, 3075
 - e_pos, 3076
 - e_type, 3075
 - end, 3076
- __gnu_pbds::trie_tag, 3076
- __gnu_pbds::trivial_iterator_tag, 3080
- __gnu_sequential, 476
- __init_winner
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 946
- __inner_product_selector
 - __gnu_parallel::_inner_product_selector< _It, _It2, _Tp >, 802
- __inplace_stable_sort
 - std, 594
- __insert_start
 - __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 944
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 946
 - __gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 948
- __int_traits
 - __gnu_cxx, 404
- __invoke
 - Utilities, 306
- __is_sorted
 - __gnu_parallel, 442
- __lg
 - std, 594
- __median
 - SGI, 156
- __median_of_three_iterators
 - __gnu_parallel, 443
- __merge_adaptive
 - std, 594

- __merge_advance
 - __gnu_parallel, 443
- __merge_advance_movc
 - __gnu_parallel, 444
- __merge_advance_usual
 - __gnu_parallel, 445
- __merge_without_buffer
 - std, 594
- __move_median_to_first
 - std, 595
- __move_merge
 - std, 595
- __move_merge_adaptive
 - std, 595
- __move_merge_adaptive_backward
 - std, 595
- __multiline
 - std::regex_constants, 727
- __new_val
 - __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 828
 - __gnu_parallel::__replace_selector< _It, _Tp >, 830
- __num_bitmaps
 - __gnu_cxx::__detail, 415
- __num_blocks
 - __gnu_cxx::__detail, 415
- __num_get_type
 - std::basic_ios< _CharT, _Traits >, 1239
 - std::basic_ofstream< _CharT, _Traits >, 1391
 - std::basic_ostream< _CharT, _Traits >, 1424
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1457
- __num_put_type
 - std::basic_ifstream< _CharT, _Traits >, 1199
 - std::basic_ios< _CharT, _Traits >, 1239
 - std::basic_istream< _CharT, _Traits >, 1311
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1351
- __parallel_merge_advance
 - __gnu_parallel, 445, 446
- __parallel_nth_element
 - __gnu_parallel, 446
- __parallel_partial_sort
 - __gnu_parallel, 447
- __parallel_partial_sum
 - __gnu_parallel, 447
- __parallel_partial_sum_basecase
 - __gnu_parallel, 448
- __parallel_partial_sum_linear
 - __gnu_parallel, 448
- __parallel_partition
 - __gnu_parallel, 449
- __parallel_random_shuffle
 - __gnu_parallel, 449
- __parallel_random_shuffle_drs
 - __gnu_parallel, 449
- __parallel_random_shuffle_drs_pu
 - __gnu_parallel, 450
- __parallel_sort
 - __gnu_parallel, 450–455
- __parallel_sort_qs
 - __gnu_parallel, 455
- __parallel_sort_qs_conquer
 - __gnu_parallel, 456
- __parallel_sort_qs_divide
 - __gnu_parallel, 456
- __parallel_sort_qsb
 - __gnu_parallel, 457
- __parallel_unique_copy
 - __gnu_parallel, 457, 458
- __partition
 - std, 595, 596
- __polynomial
 - std::regex_constants, 727
- __ptr_rebind
 - std, 590
- __qsb_conquer
 - __gnu_parallel, 458
- __qsb_divide
 - __gnu_parallel, 459
- __qsb_local_sort_with_helping
 - __gnu_parallel, 459
- __random_number_pow2
 - __gnu_parallel, 459
- __rd_log2
 - __gnu_parallel, 460
- __replace_if_selector
 - __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 828
- __replace_selector
 - __gnu_parallel::__replace_selector< _It, _Tp >, 829
- __resize_and_overwrite
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519
 - std::basic_string< _CharT, _Traits, _Alloc >, 1606
- __reverse
 - std, 596
- __rotate
 - std, 596, 597
- __rotate_adaptive
 - std, 597
- __round_up_to_pow2
 - __gnu_parallel, 460
- __sample
 - std, 597
- __search_n_aux
 - std, 598
- __search_template

- __gnu_parallel, 460
- __sequential_multiway_merge
 - __gnu_parallel, 461
- __sequential_random_shuffle
 - __gnu_parallel, 461
- __shrink
 - __gnu_parallel, 462
- __shrink_and_double
 - __gnu_parallel, 462
- __stable_partition_adaptive
 - std, 598
- __static_pointer_cast
 - __gnu_cxx, 404
- __streambuf_type
 - std::basic_streambuf< _CharT, _Traits >, 1497
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3260
- __umap_traits
 - std, 590
- __ummap_traits
 - std, 590
- __umset_traits
 - std, 590
- __uset_traits
 - std, 590
- __valid_range
 - __gnu_debug, 423
- __verbose_terminate_handler
 - Exceptions, 135
- __versa_string
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 838–841
- __yield
 - __gnu_parallel, 463
- ~LoserTreeBase
 - __gnu_parallel::LoserTreeBase< _Tp, _Compare >, 948
- ~RestrictedBoundedConcurrentQueue
 - __gnu_parallel::RestrictedBoundedConcurrentQueue< _Tp >, 985
- ~Safe_sequence_base
 - __gnu_debug::Safe_sequence_base, 1026
- ~Safe_unordered_container_base
 - __gnu_debug::Safe_unordered_container_base, 1032
- ~__versa_string
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
- ~any
 - std::experimental::fundamentals_v1::any, 1085
- ~auto_ptr
 - std::auto_ptr< _Tp >, 1102
- ~basic_filebuf
 - std::basic_filebuf< _CharT, _Traits >, 1123
- ~basic_fstream
 - std::basic_fstream< _CharT, _Traits >, 1151
- ~basic_ifstream
 - std::basic_ifstream< _CharT, _Traits >, 1202
- ~basic_ios
 - std::basic_ios< _CharT, _Traits >, 1242
- ~basic_iostream
 - std::basic_iostream< _CharT, _Traits >, 1267
- ~basic_istream
 - std::basic_istream< _CharT, _Traits >, 1314
- ~basic_istreamstream
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1354
- ~basic_ofstream
 - std::basic_ofstream< _CharT, _Traits >, 1394
- ~basic_ostream
 - std::basic_ostream< _CharT, _Traits >, 1427
- ~basic_ostringstream
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1460
- ~basic_regex
 - std::basic_regex< _Ch_type, _Rx_traits >, 1489
- ~basic_streambuf
 - std::basic_streambuf< _CharT, _Traits >, 1498
- ~basic_string
 - std::basic_string< _CharT, _Traits, _Alloc >, 1602, 1606
- ~basic_stringstream
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1729
- ~collate
 - std::__cxx11::collate< _CharT >, 1904
- ~ctype
 - std::ctype< char >, 1969
 - std::ctype< wchar_t >, 1998
- ~deque
 - std::deque< _Tp, _Alloc >, 2070
- ~facet
 - std::locale::facet, 2165
- ~forward_list
 - std::forward_list< _Tp, _Alloc >, 2183
- ~gslice
 - Numeric Arrays, 226
- ~ios_base
 - std::ios_base, 2300
- ~list
 - std::list< _Tp, _Alloc >, 2372
- ~locale
 - std::locale, 2393
- ~map
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2426
- ~match_results
 - std::match_results< _Bi_iter, _Alloc >, 2448
- ~messages
 - std::messages< _CharT >, 2469

- ~money_get
 - std::money_get< _CharT, _InIter >, 2480
- ~money_put
 - std::money_put< _CharT, _OutIter >, 2485
- ~moneypunct
 - std::moneypunct< _CharT, _Intl >, 2491
- ~multimap
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2520
- ~multiset
 - std::multiset< _Key, _Compare, _Alloc >, 2544
- ~num_get
 - std::num_get< _CharT, _InIter >, 2578
- ~num_put
 - std::num_put< _CharT, _OutIter >, 2595
- ~numpunct
 - std::numpunct< _CharT >, 2686
- ~sentry
 - std::basic_ostream< _CharT, _Traits >::sentry, 2884
- ~set
 - std::set< _Key, _Compare, _Alloc >, 2895
- ~stdio_filebuf
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2948
- ~temporary_buffer
 - __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3004
- ~time_get
 - std::time_get< _CharT, _InIter >, 3019
- ~time_put
 - std::time_put< _CharT, _OutIter >, 3042
- ~type_info
 - __gnu_debug::type_info, 3090
 - std::type_info, 3091
- ~unique_ptr
 - std::unique_ptr< _Tp, _Dp >, 3109
 - std::unique_ptr< _Tp[], _Dp >, 3115
- ~vector
 - std::vector< _Tp, _Alloc >, 3240
- a
 - std::extreme_value_distribution< _RealType >, 2161
 - std::weibull_distribution< _RealType >, 3278
- a_const_iterator
 - __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3073
- abi, 476
- abs
 - Complex Numbers, 186
 - math.h, 3635
 - std, 599
 - stdlib.h, 3695
 - Time, 376
- access_traits
 - __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3065
 - __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3073
- accumulate
 - Generalized Numeric operations, 6, 7
- accumulate_minimal_n
 - __gnu_parallel::Settings, 1039
- acosh
 - std, 599
- Adaptors for pointers to functions, 318
 - ptr_fun, 318
- Adaptors for pointers to members, 319
- add_lvalue_reference_t
 - Metaprogramming, 360
- add_pointer_t
 - Metaprogramming, 360
- add_rvalue_reference_t
 - Metaprogramming, 360
- addressof
 - Utilities, 306
- adjacent_difference
 - Generalized Numeric operations, 7, 8
- adjacent_difference_minimal_n
 - __gnu_parallel::Settings, 1039
- adjacent_find
 - Non-Mutating, 45
- adjustfield
 - std::basic_fstream< _CharT, _Traits >, 1186
 - std::basic_ifstream< _CharT, _Traits >, 1230
 - std::basic_ios< _CharT, _Traits >, 1254
 - std::basic_iostream< _CharT, _Traits >, 1301
 - std::basic_istream< _CharT, _Traits >, 1340
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1381
 - std::basic_ofstream< _CharT, _Traits >, 1415
 - std::basic_ostream< _CharT, _Traits >, 1446
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1479
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1764
 - std::ios_base, 2305
- adopt_lock
 - Mutexes, 124
- advance
 - std, 599
- airy_ai
 - Mathematical Special Functions, 196
- airy_aif
 - Mathematical Special Functions, 196
- airy_ail
 - Mathematical Special Functions, 196
- airy_bi
 - Mathematical Special Functions, 196

- airy_bif
 - Mathematical Special Functions, [196](#)
- airy_bil
 - Mathematical Special Functions, [197](#)
- algo.h, [3650](#)
- algbase.h, [3660](#)
- algorithm, [3283](#), [3285](#)
- algorithmfwd.h, [3294](#), [3299](#)
- Algorithms, [4](#)
- align
 - Memory, [327](#)
- align.h, [3307](#)
- aligned_buffer.h, [3543](#)
- aligned_storage_t
 - Metaprogramming, [360](#)
- alignment_value
 - Metaprogramming, [363](#)
- all
 - std::bitset< _Nb >, [1815](#)
 - std::locale, [2396](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2109](#)
- all_of
 - Non-Mutating, [46](#)
- alloc_traits.h, [3308](#)
- allocate
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [759](#), [760](#)
 - std::allocator_traits< _Alloc >, [1065](#)
 - std::allocator_traits< allocator< _Tp > >, [1070](#)
 - std::allocator_traits< allocator< void > >, [1075](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1079](#)
- allocate_at_least
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [760](#)
 - std::allocator_traits< _Alloc >, [1066](#)
 - std::allocator_traits< allocator< _Tp > >, [1071](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1080](#)
- allocate_shared
 - Pointer Abstractions, [338](#)
- allocated_ptr.h, [3308](#)
- allocator.h, [3308](#)
- allocator_type
 - __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, [3073](#)
 - std::allocator_traits< _Alloc >, [1063](#)
 - std::allocator_traits< allocator< _Tp > >, [1069](#)
 - std::allocator_traits< allocator< void > >, [1074](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1078](#)
 - std::set< _Key, _Compare, _Alloc >, [2891](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3124](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3153](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3178](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3202](#)
- Allocators, [333](#)
 - __allocator_base, [334](#)
 - operator==, [334](#)
- alpha
 - std::gamma_distribution< _RealType >, [2218](#)
- any, [3285](#)
 - std::bitset< _Nb >, [1815](#)
 - std::experimental::fundamentals_v1::any, [1084](#), [1085](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2109](#)
- any_cast
 - Type-safe container of any type, [295–297](#)
- any_of
 - Non-Mutating, [46](#)
- app
 - std::basic_fstream< _CharT, _Traits >, [1187](#)
 - std::basic_ifstream< _CharT, _Traits >, [1230](#)
 - std::basic_ios< _CharT, _Traits >, [1254](#)
 - std::basic_iostream< _CharT, _Traits >, [1301](#)
 - std::basic_istream< _CharT, _Traits >, [1341](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1381](#)
 - std::basic_ofstream< _CharT, _Traits >, [1415](#)
 - std::basic_ostream< _CharT, _Traits >, [1446](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1479](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1764](#)
 - std::ios_base, [2305](#)
- append
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [841–844](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1519–1521](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1606–1611](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2109](#)
- apply
 - Numeric Arrays, [226](#), [227](#)
- apply_generator
 - __gnu_cxx::typelist, [416](#)
- arg
 - Complex Numbers, [186](#)
 - std::tr1, [737](#)
- argument_type
 - __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, [1783](#)
 - __gnu_cxx::select1st< _Pair >, [2881](#)
 - __gnu_cxx::select2nd< _Pair >, [2881](#)

- `__gnu_cxx::subtractive_rng`, 3000
- `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`, 3095
- `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 766
- `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 767
- `__gnu_parallel::__unary_negate< _Predicate, argument_type >`, 834
- `std::binder1st< _Operation >`, 1797
- `std::binder2nd< _Operation >`, 1799
- `std::const_mem_fun_ref_t< _Ret, _Tp >`, 1925
- `std::const_mem_fun_t< _Ret, _Tp >`, 1926
- `std::hash< __gnu_cxx::throw_value_limit >`, 2247
- `std::hash< __gnu_cxx::throw_value_random >`, 2248
- `std::logical_not< _Tp >`, 2402
- `std::logical_not< void >`, 2403
- `std::mem_fun_ref_t< _Ret, _Tp >`, 2459
- `std::mem_fun_t< _Ret, _Tp >`, 2460
- `std::negate< _Tp >`, 2560
- `std::negate< void >`, 2561
- `std::pointer_to_unary_function< _Arg, _Result >`, 2772
- `std::unary_function< _Arg, _Result >`, 3096
- `std::unary_negate< _Predicate >`, 3097
- Arithmetic Function Object Classes, 320
- array, 3286, 3287
- Array creation functions, 288
 - `make_array`, 288
 - `to_array`, 288
- assertions.h, 3514
- assign
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 844–847
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1522, 1523
 - `std::basic_regex< _Ch_type, _Rx_traits >`, 1489–1491
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1612–1618
 - `std::deque< _Tp, _Alloc >`, 2074
 - `std::error_condition`, 2152
 - `std::forward_list< _Tp, _Alloc >`, 2183, 2184
 - `std::list< _Tp, _Alloc >`, 2372, 2373
 - `std::vector< _Tp, _Alloc >`, 3240, 3241
- `assoc_container.hpp`, 3548
- `assoc_laguerre`
 - Mathematical Special Functions, 197
 - TR1 Mathematical Special Functions, 253
- `assoc_laguerref`
 - Mathematical Special Functions, 198
- `assoc_laguerrel`
 - Mathematical Special Functions, 198
- `assoc_legendre`
 - Mathematical Special Functions, 198
 - TR1 Mathematical Special Functions, 253
- `assoc_legendref`
 - Mathematical Special Functions, 199
- `assoc_legendrel`
 - Mathematical Special Functions, 199
- Associative, 127
- `assume_aligned`
 - Memory, 328
- async
 - Futures, 120
- at
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 848
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1524–1526
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1618–1620
 - `std::deque< _Tp, _Alloc >`, 2075
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2426
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3128–3130
 - `std::vector< _Tp, _Alloc >`, 3241, 3242
- ate
 - `std::basic_fstream< _CharT, _Traits >`, 1187
 - `std::basic_ifstream< _CharT, _Traits >`, 1230
 - `std::basic_ios< _CharT, _Traits >`, 1254
 - `std::basic_iostream< _CharT, _Traits >`, 1301
 - `std::basic_istream< _CharT, _Traits >`, 1341
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1381
 - `std::basic_ofstream< _CharT, _Traits >`, 1415
 - `std::basic_ostream< _CharT, _Traits >`, 1446
 - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1480
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1764
 - `std::ios_base`, 2305
- atomic, 3287
- `atomic_base.h`, 3309
- `atomic_bool`
 - Atomics, 110
- ATOMIC_BOOL_LOCK_FREE
 - Atomics, 109
- `atomic_char`
 - Atomics, 110
- `atomic_char16_t`
 - Atomics, 110
- `atomic_char32_t`
 - Atomics, 110
- `atomic_compare_exchange_strong`

Pointer Abstractions, [339](#)
atomic_compare_exchange_strong_explicit
Pointer Abstractions, [340](#)
atomic_compare_exchange_weak
Pointer Abstractions, [341](#)
atomic_compare_exchange_weak_explicit
Pointer Abstractions, [342](#)
atomic_exchange
Pointer Abstractions, [343](#)
atomic_exchange_explicit
Pointer Abstractions, [343](#), [344](#)
atomic_futex.h, [3309](#)
atomic_int
Atomics, [110](#)
atomic_int16_t
Atomics, [110](#)
atomic_int32_t
Atomics, [110](#)
atomic_int64_t
Atomics, [110](#)
atomic_int8_t
Atomics, [111](#)
atomic_int_fast16_t
Atomics, [111](#)
atomic_int_fast32_t
Atomics, [111](#)
atomic_int_fast64_t
Atomics, [111](#)
atomic_int_fast8_t
Atomics, [111](#)
atomic_int_least16_t
Atomics, [111](#)
atomic_int_least32_t
Atomics, [111](#)
atomic_int_least64_t
Atomics, [112](#)
atomic_int_least8_t
Atomics, [112](#)
atomic_intmax_t
Atomics, [112](#)
atomic_intptr_t
Atomics, [112](#)
atomic_is_lock_free
Pointer Abstractions, [344](#), [345](#)
atomic_llong
Atomics, [112](#)
atomic_load
Pointer Abstractions, [345](#)
atomic_load_explicit
Pointer Abstractions, [346](#)
atomic_lockfree_defines.h, [3310](#)
atomic_long
Atomics, [112](#)
atomic_ptrdiff_t

Atomics, [112](#)
atomic_schar
Atomics, [113](#)
atomic_short
Atomics, [113](#)
atomic_size_t
Atomics, [113](#)
atomic_store
Pointer Abstractions, [346](#), [347](#)
atomic_store_explicit
Pointer Abstractions, [347](#), [348](#)
atomic_timed_wait.h, [3310](#)
atomic_uchar
Atomics, [113](#)
atomic_uint
Atomics, [113](#)
atomic_uint16_t
Atomics, [113](#)
atomic_uint32_t
Atomics, [113](#)
atomic_uint64_t
Atomics, [114](#)
atomic_uint8_t
Atomics, [114](#)
atomic_uint_fast16_t
Atomics, [114](#)
atomic_uint_fast32_t
Atomics, [114](#)
atomic_uint_fast64_t
Atomics, [114](#)
atomic_uint_fast8_t
Atomics, [114](#)
atomic_uint_least16_t
Atomics, [114](#)
atomic_uint_least32_t
Atomics, [115](#)
atomic_uint_least64_t
Atomics, [115](#)
atomic_uint_least8_t
Atomics, [115](#)
atomic_uintmax_t
Atomics, [115](#)
atomic_uintptr_t
Atomics, [115](#)
atomic_ullong
Atomics, [115](#)
atomic_ulong
Atomics, [115](#)
atomic_ushort
Atomics, [116](#)
atomic_wait.h, [3310](#)
atomic_wchar_t
Atomics, [116](#)
atomic_word.h, [3724](#)

atomicity.h, [3543](#)

Atomics, [105](#)

- atomic_bool, [110](#)
- ATOMIC_BOOL_LOCK_FREE, [109](#)
- atomic_char, [110](#)
- atomic_char16_t, [110](#)
- atomic_char32_t, [110](#)
- atomic_int, [110](#)
- atomic_int16_t, [110](#)
- atomic_int32_t, [110](#)
- atomic_int64_t, [110](#)
- atomic_int8_t, [111](#)
- atomic_int_fast16_t, [111](#)
- atomic_int_fast32_t, [111](#)
- atomic_int_fast64_t, [111](#)
- atomic_int_fast8_t, [111](#)
- atomic_int_least16_t, [111](#)
- atomic_int_least32_t, [111](#)
- atomic_int_least64_t, [112](#)
- atomic_int_least8_t, [112](#)
- atomic_intmax_t, [112](#)
- atomic_intptr_t, [112](#)
- atomic_llong, [112](#)
- atomic_long, [112](#)
- atomic_ptrdiff_t, [112](#)
- atomic_schar, [113](#)
- atomic_short, [113](#)
- atomic_size_t, [113](#)
- atomic_uchar, [113](#)
- atomic_uint, [113](#)
- atomic_uint16_t, [113](#)
- atomic_uint32_t, [113](#)
- atomic_uint64_t, [114](#)
- atomic_uint8_t, [114](#)
- atomic_uint_fast16_t, [114](#)
- atomic_uint_fast32_t, [114](#)
- atomic_uint_fast64_t, [114](#)
- atomic_uint_fast8_t, [114](#)
- atomic_uint_least16_t, [114](#)
- atomic_uint_least32_t, [115](#)
- atomic_uint_least64_t, [115](#)
- atomic_uint_least8_t, [115](#)
- atomic_uintmax_t, [115](#)
- atomic_uintptr_t, [115](#)
- atomic_ullong, [115](#)
- atomic_ulong, [115](#)
- atomic_ushort, [116](#)
- atomic_wchar_t, [116](#)
- kill_dependency, [116](#)
- memory_order, [116](#)

auto_ptr

std::auto_ptr<_Tp>, [1102](#), [1103](#)

auto_ptr.h, [3291](#)

awk

std::regex_constants, [728](#)

b

std::extreme_value_distribution<_RealType>, [2161](#)
std::weibull_distribution<_RealType>, [3278](#)

back

__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>, [848](#), [849](#)
__gnu_debug::basic_string<_CharT, _Traits, _Allocator>, [1527](#)
std::basic_string<_CharT, _Traits, _Alloc>, [1620](#)
std::deque<_Tp, _Alloc>, [2076](#)
std::list<_Tp, _Alloc>, [2373](#)
std::queue<_Tp, _Sequence>, [2795](#)
std::vector<_Tp, _Alloc>, [3242](#)

back_insert_iterator

std::back_insert_iterator<_Container>, [1107](#)

back_inserter

Iterators, [176](#)

backward_warning.h, [3291](#)

bad

std::basic_fstream<_CharT, _Traits>, [1151](#)
std::basic_ifstream<_CharT, _Traits>, [1202](#)
std::basic_ios<_CharT, _Traits>, [1243](#)
std::basic_iostream<_CharT, _Traits>, [1267](#)
std::basic_istream<_CharT, _Traits>, [1314](#)
std::basic_istreamstream<_CharT, _Traits, _Alloc>, [1355](#)
std::basic_ofstream<_CharT, _Traits>, [1394](#)
std::basic_ostream<_CharT, _Traits>, [1427](#)
std::basic_ostreamstream<_CharT, _Traits, _Alloc>, [1460](#)
std::basic_stringstream<_CharT, _Traits, _Alloc>, [1730](#)

badbit

std::basic_fstream<_CharT, _Traits>, [1187](#)
std::basic_ifstream<_CharT, _Traits>, [1230](#)
std::basic_ios<_CharT, _Traits>, [1255](#)
std::basic_iostream<_CharT, _Traits>, [1302](#)
std::basic_istream<_CharT, _Traits>, [1341](#)
std::basic_istreamstream<_CharT, _Traits, _Alloc>, [1382](#)
std::basic_ofstream<_CharT, _Traits>, [1415](#)
std::basic_ostream<_CharT, _Traits>, [1447](#)
std::basic_ostreamstream<_CharT, _Traits, _Alloc>, [1480](#)
std::basic_stringstream<_CharT, _Traits, _Alloc>, [1764](#)
std::ios_base, [2305](#)

balanced_quicksort.h, [3662](#)

barrier, [3293](#)

base

__gnu_debug::__Safe_iterator<_Iterator, _Sequence, _Category>, [997](#)

- `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1011
- `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2093
- `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`, 2281
- `std::reverse_iterator< _Iterator >`, 2855
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2926
- Base and Implementation Classes, 278
 - `_Opcode`, 279
- Base and Policy Classes, 146–148
- `base.h`, 3662
- basefield
 - `std::basic_fstream< _CharT, _Traits >`, 1187
 - `std::basic_ifstream< _CharT, _Traits >`, 1230
 - `std::basic_ios< _CharT, _Traits >`, 1255
 - `std::basic_iostream< _CharT, _Traits >`, 1302
 - `std::basic_istream< _CharT, _Traits >`, 1341
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1382
 - `std::basic_ofstream< _CharT, _Traits >`, 1415
 - `std::basic_ostream< _CharT, _Traits >`, 1447
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1480
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1764
 - `std::ios_base`, 2305
- basic
 - `std::regex_constants`, 728
- `basic_file.h`, 3724
- basic_filebuf
 - `std::basic_filebuf< _CharT, _Traits >`, 1123
- basic_fstream
 - `std::basic_fstream< _CharT, _Traits >`, 1150, 1151
- basic_ifstream
 - `std::basic_ifstream< _CharT, _Traits >`, 1201, 1202
- basic_ios
 - `std::basic_ios< _CharT, _Traits >`, 1242, 1243
- `basic_ios.h`, 3310
- `basic_ios.tcc`, 3311
- basic_iostream
 - `std::basic_iostream< _CharT, _Traits >`, 1267
- basic_istream
 - `std::basic_istream< _CharT, _Traits >`, 1314
- basic_istreamstream
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1354
- `basic_iterator.h`, 3663
- basic_ofstream
 - `std::basic_ofstream< _CharT, _Traits >`, 1393, 1394
- basic_ostream
 - `std::basic_ostream< _CharT, _Traits >`, 1427
- basic_ostringstream
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1459
- basic_regex
 - `std::basic_regex< _Ch_type, _Rx_traits >`, 1486–1488
- basic_streambuf
 - `std::basic_streambuf< _CharT, _Traits >`, 1498
- basic_string
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1599–1605
- `basic_string.h`, 3311
- `basic_string.tcc`, 3314
- basic_stringbuf
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1704, 1705
- basic_stringstream
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1729
- before
 - `__gnu_debug::type_info`, 3090
 - `std::type_info`, 3091
- before_begin
 - `std::forward_list< _Tp, _Alloc >`, 2184
- beg
 - `std::basic_fstream< _CharT, _Traits >`, 1187
 - `std::basic_ifstream< _CharT, _Traits >`, 1230
 - `std::basic_ios< _CharT, _Traits >`, 1255
 - `std::basic_iostream< _CharT, _Traits >`, 1302
 - `std::basic_istream< _CharT, _Traits >`, 1341
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1382
 - `std::basic_ofstream< _CharT, _Traits >`, 1415
 - `std::basic_ostream< _CharT, _Traits >`, 1447
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1480
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1764
 - `std::ios_base`, 2305
- begin
 - `__gnu_cxx::Temporary_buffer< _ForwardIterator, _Tp >`, 1048
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 849
 - `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`, 3005
 - `__gnu_parallel::PseudoSequence< _Tp, _DifferenceTp >`, 978
 - `__gnu_pbds::sample_trie_access_traits`, 2874
 - `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >`, 3075
- Filesystem, 167
- std, 600
- `std::Temporary_buffer< _ForwardIterator, _Tp >`,

- 1050
- std::basic_string< _CharT, _Traits, _Alloc >, 1621
- std::deque< _Tp, _Alloc >, 2076
- std::forward_list< _Tp, _Alloc >, 2185
- std::list< _Tp, _Alloc >, 2373, 2374
- std::map< _Key, _Tp, _Compare, _Alloc >, 2426, 2427
- std::match_results< _Bi_iter, _Alloc >, 2449
- std::multimap< _Key, _Tp, _Compare, _Alloc >, 2520
- std::multiset< _Key, _Compare, _Alloc >, 2544
- std::set< _Key, _Compare, _Alloc >, 2896
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3130, 3131
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3158
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3183
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3207
- std::valarray< _Tp >, 3227
- std::vector< _Tp, _Alloc >, 3242
- Bernoulli Distributions, 239
 - operator<<, 239, 240
 - operator>>, 240
- bernoulli_distribution
 - std::bernoulli_distribution, 1769
- beta
 - Mathematical Special Functions, 199
 - std::gamma_distribution< _RealType >, 2218
 - TR1 Mathematical Special Functions, 253
- betaf
 - Mathematical Special Functions, 200
- betal
 - Mathematical Special Functions, 200
- bin_search_tree.hpp, 3549
- binary
 - std::basic_fstream< _CharT, _Traits >, 1187
 - std::basic_ifstream< _CharT, _Traits >, 1231
 - std::basic_ios< _CharT, _Traits >, 1255
 - std::basic_iostream< _CharT, _Traits >, 1302
 - std::basic_istream< _CharT, _Traits >, 1341
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1382
 - std::basic_ofstream< _CharT, _Traits >, 1415
 - std::basic_ostream< _CharT, _Traits >, 1447
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1480
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1764
 - std::ios_base, 2306
- Binary Search, 85
 - binary_search, 86, 87
 - equal_range, 87, 88
 - lower_bound, 88, 89
 - upper_bound, 89, 90
- binary_heap.hpp, 3552
- binary_heap_const_iterator_
 - gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1790
- binary_heap_point_const_iterator_
 - gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1794
- binary_search
 - Binary Search, 86, 87
- bind
 - Binder Classes, 322
- bind1st
 - Binder Classes, 322
- bind2nd
 - Binder Classes, 322
- Binder Classes, 320
 - bind, 322
 - bind1st, 322
 - bind2nd, 322
- binders.h, 3314
- binomial_heap.hpp, 3566
- binomial_heap_base.hpp, 3567
- bit, 3294
- Bit manipulation, 182
- bitmap_allocator.h, 3544
 - _BALLOC_ALIGN_BYTES, 3545
- bitset, 3480, 3481
 - std::bitset< _Nb >, 1813, 1814
- bool_set, 3704
 - std::tr2::bool_set, 1822
- bool_set.tcc, 3705
- boolalpha
 - std, 601
 - std::basic_fstream< _CharT, _Traits >, 1187
 - std::basic_ifstream< _CharT, _Traits >, 1231
 - std::basic_ios< _CharT, _Traits >, 1255
 - std::basic_iostream< _CharT, _Traits >, 1302
 - std::basic_istream< _CharT, _Traits >, 1341
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1382
 - std::basic_ofstream< _CharT, _Traits >, 1415
 - std::basic_ostream< _CharT, _Traits >, 1447
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1480
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1765
 - std::ios_base, 2306
- Boolean Operations Classes, 323
- boost_concept_check.h, 3315
- Branch-Based, 145
- branch_policy.hpp, 3567
- bucket

- `__gnu_debug::Safe_local_iterator< _Iterator, _UContainer >`, 1011
 - bucket_count
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3131
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3158
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3184
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3208
 - c
 - `std::queue< _Tp, _Sequence >`, 2796
 - c++0x_warning.h, 3315
 - c++allocator.h, 3724
 - c++config.h, 3725
 - c++io.h, 3731
 - c++locale.h, 3731
 - c++locale_internal.h, 3731
 - c_str
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 849
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1621
 - cache_line_size
 - `__gnu_parallel::Settings`, 1039
 - call_once
 - Mutexes, 123
 - `std::once_flag`, 2695
 - capacity
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 849
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1527
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1621, 1622
 - `std::vector< _Tp, _Alloc >`, 3243
 - cassert, 3482
 - cast.h, 3545
 - category
 - `std::error_code`, 2150
 - `std::error_condition`, 2152
 - `std::locale`, 2390
 - cbefore_begin
 - `std::forward_list< _Tp, _Alloc >`, 2185
 - cbegin
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 849
 - std, 601
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1622
 - `std::deque< _Tp, _Alloc >`, 2076
 - `std::forward_list< _Tp, _Alloc >`, 2185
 - `std::list< _Tp, _Alloc >`, 2374
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2427
 - `std::match_results< _Bi_iter, _Alloc >`, 2449
 - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2520
 - `std::multiset< _Key, _Compare, _Alloc >`, 2544
 - `std::set< _Key, _Compare, _Alloc >`, 2896
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3131
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3159
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3184
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3208
 - `std::vector< _Tp, _Alloc >`, 3243
- cc
 - cc_hash_max_collision_check_resize_trigger
 - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`, 1828
 - cc_hash_max_collision_check_resize_trigger_imp.hpp, 3585
 - cc_hash_table
 - `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Re-size_Policy, Store_Hash, _Alloc >`, 1832–1835
- cc_ht_map.hpp, 3567
- ccomplex, 3482
- cctype, 3482
- ceil
 - chrono.h, 3320
 - Time, 376, 377
- cend
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 850
 - std, 601
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1622
 - `std::deque< _Tp, _Alloc >`, 2076
 - `std::forward_list< _Tp, _Alloc >`, 2185
 - `std::list< _Tp, _Alloc >`, 2374
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2427
 - `std::match_results< _Bi_iter, _Alloc >`, 2449
 - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2520
 - `std::multiset< _Key, _Compare, _Alloc >`, 2544
 - `std::set< _Key, _Compare, _Alloc >`, 2896
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3132
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3159
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3184
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3208
 - `std::vector< _Tp, _Alloc >`, 3243
- cerr
 - std, 657

- cerrno, [3483](#)
- cfenv, [3483](#)
- cfloat, [3483](#)
- char_traits.h, [3316](#)
- char_type
 - std::__ctype_abstract_base<_CharT>, [776](#)
 - std::__cxx11::collate<_CharT>, [1903](#)
 - std::__cxx11::collate_byname<_CharT>, [1908](#)
 - std::basic_ios<_CharT, _Traits>, [1240](#)
 - std::basic_streambuf<_CharT, _Traits>, [1497](#)
 - std::ctype<char>, [1968](#)
 - std::ctype<wchar_t>, [1998](#)
 - std::istreambuf_iterator<_CharT, _Traits>, [2334](#)
 - std::messages<_CharT>, [2468](#)
 - std::money_get<_CharT, _InIter>, [2480](#)
 - std::money_put<_CharT, _OutIter>, [2484](#)
 - std::moneypunct<_CharT, _Intl>, [2490](#)
 - std::num_get<_CharT, _InIter>, [2578](#)
 - std::num_put<_CharT, _OutIter>, [2595](#)
 - std::numpunct<_CharT>, [2685](#)
 - std::ostream_iterator<_Tp, _CharT, _Traits>, [2698](#)
 - std::ostreambuf_iterator<_CharT, _Traits>, [2702](#)
 - std::time_get<_CharT, _InIter>, [3019](#)
 - std::time_put<_CharT, _OutIter>, [3042](#)
 - std::wbuffer_convert<_Codecvt, _Elem, _Tr>, [3260](#)
- charconv, [3484](#)
- charconv.h, [3316](#)
- chars_format
 - std, [592](#)
- checkers.h, [3663](#)
- chrono, [3485](#), [3487](#)
- chrono.h, [3316](#)
 - ceil, [3320](#)
- chrono_io.h, [3321](#)
- cin
 - std, [657](#)
- cinttypes, [3488](#)
- ciso646, [3488](#)
- clamp
 - Sorting, [65](#)
- classic
 - std::locale, [2393](#)
- classic_table
 - std::ctype<char>, [1969](#)
 - std::ctype_byname<char>, [2027](#)
- clear
 - __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>, [850](#)
 - std::basic_fstream<_CharT, _Traits>, [1152](#)
 - std::basic_ifstream<_CharT, _Traits>, [1203](#)
 - std::basic_ios<_CharT, _Traits>, [1243](#)
 - std::basic_iostream<_CharT, _Traits>, [1268](#)
 - std::basic_istream<_CharT, _Traits>, [1314](#)
 - std::basic_istreambuf_iterator<_CharT, _Traits, _Alloc>, [1355](#)
 - std::basic_ofstream<_CharT, _Traits>, [1395](#)
 - std::basic_ostream<_CharT, _Traits>, [1427](#)
 - std::basic_ostringstream<_CharT, _Traits, _Alloc>, [1460](#)
 - std::basic_string<_CharT, _Traits, _Alloc>, [1622](#)
 - std::basic_stringstream<_CharT, _Traits, _Alloc>, [1730](#)
 - std::deque<_Tp, _Alloc>, [2076](#)
 - std::error_condition, [2152](#)
 - std::experimental::fundamentals_v1::any, [1085](#)
 - std::forward_list<_Tp, _Alloc>, [2185](#)
 - std::list<_Tp, _Alloc>, [2374](#)
 - std::map<_Key, _Tp, _Compare, _Alloc>, [2427](#)
 - std::multimap<_Key, _Tp, _Compare, _Alloc>, [2521](#)
 - std::multiset<_Key, _Compare, _Alloc>, [2544](#)
 - std::set<_Key, _Compare, _Alloc>, [2896](#)
 - std::tr2::dynamic_bitset<_WordT, _Alloc>, [2109](#)
 - std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>, [3132](#)
 - std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>, [3160](#)
 - std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>, [3185](#)
 - std::unordered_set<_Value, _Hash, _Pred, _Alloc>, [3209](#)
 - std::vector<_Tp, _Alloc>, [3243](#)
- climits, [3488](#), [3489](#)
- locale, [3489](#)
- clock_cast
 - Time, [377](#)
- clog
 - std, [657](#)
- close
 - __gnu_cxx::enc_filebuf<_CharT>, [2123](#)
 - __gnu_cxx::stdio_filebuf<_CharT, _Traits>, [2948](#)
 - std::basic_filebuf<_CharT, _Traits>, [1124](#)
 - std::basic_fstream<_CharT, _Traits>, [1152](#)
 - std::basic_ifstream<_CharT, _Traits>, [1203](#)
 - std::basic_ofstream<_CharT, _Traits>, [1395](#)
- cmath, [3489](#), [3495](#), [3496](#)
- cmp_fn
 - __gnu_pbds::tree<_Key, _Mapped, _Cmp_Fn, _Tag, _Node_Update, _Alloc>, [3052](#)
- cmp_fn_imps.hpp, [3568](#)
- code
 - std::regex_error, [2835](#)
- codecvt, [3499](#)
- codecvt.h, [3324](#)
- codecvt_specializations.h, [3545](#)
- collate
 - std::__cxx11::collate<_CharT>, [1903](#), [1904](#)

- std::locale, [2396](#)
- std::regex_constants, [728](#)
- combine
 - std::locale, [2393](#)
- common_reference_t
 - Metaprogramming, [360](#)
- common_type_t
 - Metaprogramming, [360](#)
- comp_ellint_1
 - Mathematical Special Functions, [200](#)
 - TR1 Mathematical Special Functions, [253](#)
- comp_ellint_1f
 - Mathematical Special Functions, [201](#)
- comp_ellint_1l
 - Mathematical Special Functions, [201](#)
- comp_ellint_2
 - Mathematical Special Functions, [201](#)
 - TR1 Mathematical Special Functions, [254](#)
- comp_ellint_2f
 - Mathematical Special Functions, [202](#)
- comp_ellint_2l
 - Mathematical Special Functions, [202](#)
- comp_ellint_3
 - Mathematical Special Functions, [202](#)
 - TR1 Mathematical Special Functions, [254](#)
- comp_ellint_3f
 - Mathematical Special Functions, [203](#)
- comp_ellint_3l
 - Mathematical Special Functions, [203](#)
- compare, [3735](#)
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [850–852](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1527–1533](#)
 - std::__cxx11::collate< _CharT >, [1904](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1622–1628](#)
 - std::sub_match< _Biter >, [2992](#)
- compare_three_way_result_t
 - std, [590](#)
- Comparison Classes, [323](#)
- compatibility.h, [3732](#)
- compiletime_settings.h, [3663](#)
 - _GLIBCXX_CALL, [3664](#)
 - _GLIBCXX_PARALLEL_ASSERTIONS, [3664](#)
 - _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1, [3664](#)
 - _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB, [3664](#)
 - _GLIBCXX_SCALE_DOWN_FPU, [3664](#)
 - _GLIBCXX_VERBOSE_LEVEL, [3664](#)
- complex, [3499](#), [3503](#)
 - std::complex< _Tp >, [1912](#)
- Complex Numbers, [182](#)
- abs, [186](#)
- arg, [186](#)
- conj, [186](#)
- cos, [186](#)
- cosh, [186](#)
- exp, [186](#)
- fabs, [186](#)
- log, [186](#)
- log10, [187](#)
- norm, [187](#)
- operator<<, [189](#)
- operator>>, [189](#)
- operator+, [187](#), [188](#)
- operator-, [188](#)
- operator/, [188](#), [189](#)
- operator==, [189](#)
- operator*, [187](#)
- polar, [190](#)
- pow, [190](#)
- sin, [190](#)
- sinh, [191](#)
- sqrt, [191](#)
- tan, [191](#)
- tanh, [191](#)
- complex.h, [3504](#)
- compose1
 - SGI, [158](#)
- compose2
 - SGI, [158](#)
- concept_check.h, [3325](#)
- Concepts, [315](#)
- concurrency.h, [3546](#)
- Concurrency, [117](#)
- cond_dealtor.hpp, [3572](#)
- cond_key_dtor_entry_dealtor.hpp, [3568](#)
- Condition Variables, [117](#)
 - cv_status, [118](#)
- condition_variable, [3507](#)
- conditional_t
 - Metaprogramming, [360](#)
- conf_hyperg
 - Mathematical Special Functions, [204](#)
 - std::tr1, [737](#)
- conf_hypergf
 - Mathematical Special Functions, [204](#)
 - std::tr1, [737](#)
- conf_hypergl
 - Mathematical Special Functions, [204](#)
 - std::tr1, [738](#)
- conj
 - Complex Numbers, [186](#)
- Const-propagating wrapper, [289](#)
- const_iterator
 - __gnu_pbds::trie_string_access_traits< String,

- Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3075
- std::set< _Key, _Compare, _Alloc >, 2891
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3178
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3202
- const_iterator.hpp, 3553, 3554
- const_local_iterator
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- const_pointer
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1789
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1793
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2345
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2349
 - std::allocator_traits< _Alloc >, 1063
 - std::allocator_traits< allocator< _Tp > >, 1069
 - std::allocator_traits< allocator< void > >, 1074
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
 - std::set< _Key, _Compare, _Alloc >, 2891
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- const_pointer_cast
 - Pointer Abstractions, 348
 - std, 601
- const_reference
 - __gnu_pbds::detail::bin_search_tree_const_node_iterator< Node, Const_Iterator, Iterator, _Alloc >, 1775
 - __gnu_pbds::detail::bin_search_tree_node_iterator< Node, Const_Iterator, Iterator, _Alloc >, 1779
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1789
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1793
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2345
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2349
 - std::allocator_traits< _Alloc >, 1063
 - std::allocator_traits< allocator< _Tp > >, 1069
 - std::allocator_traits< allocator< void > >, 1074
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
 - std::set< _Key, _Compare, _Alloc >, 2891
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- const_iterator
 - std::set< _Key, _Compare, _Alloc >, 2891
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3125
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3154
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- const_reverse_iterator
 - std::set< _Key, _Compare, _Alloc >, 2891
- const_void_pointer
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 758
 - std::allocator_traits< _Alloc >, 1063
 - std::allocator_traits< allocator< _Tp > >, 1069
 - std::allocator_traits< allocator< void > >, 1074
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
- constant0
 - SGI, 158
- constant1
 - SGI, 159
- constant2
 - SGI, 159
- construct
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 761
 - std::allocator_traits< _Alloc >, 1066
 - std::allocator_traits< allocator< _Tp > >, 1071
 - std::allocator_traits< allocator< void > >, 1075
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1080
- constructor_destructor_fn_imps.hpp, 3569
- constructor_destructor_no_store_hash_fn_imps.hpp, 3569
- constructor_destructor_store_hash_fn_imps.hpp, 3569
- constructors_destructor_fn_imps.hpp, 3554, 3555
- container_base_dispatch.hpp, 3572
- container_type
 - std::back_insert_iterator< _Container >, 1107
 - std::front_insert_iterator< _Container >, 2201
 - std::insert_iterator< _Container >, 2291
- Containers, 126, 145
 - contains
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2427
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2521
 - std::multiset< _Key, _Compare, _Alloc >, 2545
 - std::set< _Key, _Compare, _Alloc >, 2896

- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3132](#), [3133](#)
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3160](#)
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3185](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3209](#)
- converted
 - std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, [3281](#)
- copy
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [853](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1534](#)
 - Mutating, [23](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1628](#), [1629](#)
- copy_backward
 - Mutating, [24](#)
- copy_if
 - Mutating, [24](#)
- copy_n
 - Mutating, [25](#)
 - SGI, [159](#)
- copy_options
 - Filesystem, [167](#)
 - Filesystem TS, [286](#)
- copyable_function
 - std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV630
noexcept(_Noex)>, [1947](#), [1948](#)
- copyfmt
 - std::basic_fstream< _CharT, _Traits >, [1152](#)
 - std::basic_ifstream< _CharT, _Traits >, [1203](#)
 - std::basic_ios< _CharT, _Traits >, [1244](#)
 - std::basic_iostream< _CharT, _Traits >, [1268](#)
 - std::basic_istream< _CharT, _Traits >, [1315](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1355](#)
 - std::basic_ofstream< _CharT, _Traits >, [1395](#)
 - std::basic_ostream< _CharT, _Traits >, [1428](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1461](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1730](#)
- cos
 - Complex Numbers, [186](#)
 - math.h, [3635](#)
- cosh
 - Complex Numbers, [186](#)
 - math.h, [3635](#)
- count
 - Non-Mutating, [47](#)
- std::bitset< _Nb >, [1815](#)
- std::map< _Key, _Tp, _Compare, _Alloc >, [2428](#)
- std::multimap< _Key, _Tp, _Compare, _Alloc >, [2521](#)
- std::multiset< _Key, _Compare, _Alloc >, [2545](#)
- std::set< _Key, _Compare, _Alloc >, [2897](#)
- std::tr2::dynamic_bitset< _WordT, _Alloc >, [2109](#)
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3133](#)
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3160](#)
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3186](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3210](#)
- count_if
 - Non-Mutating, [47](#)
- count_minimal_n
 - __gnu_parallel::Settings, [1039](#)
- cout
 - std, [657](#)
- cow_string.h, [3325](#)
- cpp_type_traits.h, [3325](#)
- cpu_defines.h, [3732](#)
- cpyfunc_impl.h, [3326](#)
- crbegin
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [854](#)
 - std, [602](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1629](#), [1630](#)
 - std::deque< _Tp, _Alloc >, [2077](#)
 - std::list< _Tp, _Alloc >, [2374](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2428](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2522](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2546](#)
 - std::set< _Key, _Compare, _Alloc >, [2897](#)
 - std::vector< _Tp, _Alloc >, [3243](#)
- cref
 - std::reference_wrapper< _Tp >, [2833](#)
- cregex_token_iterator
 - Regular Expressions, [260](#)
- crend
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [854](#)
 - std, [602](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1630](#)
 - std::deque< _Tp, _Alloc >, [2077](#)
 - std::list< _Tp, _Alloc >, [2374](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2428](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2522](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2546](#)

- std::set< _Key, _Compare, _Alloc >, 2897
- std::vector< _Tp, _Alloc >, 3243
- csetjmp, 3507
- cshift
 - Numeric Arrays, 227
- csignal, 3507
- cstdalign, 3508
- cstdarg, 3508
- cstdbool, 3508, 3509
- cstddef, 3509
- cstdint, 3509
- cstdio, 3510
- cstdlib, 3510, 3511
- cstring, 3511
- csub_match
 - Regular Expressions, 260
- ctgmach, 3511
- ctime, 3512
- ctype
 - std::ctype< char >, 1969
 - std::ctype< wchar_t >, 1998
 - std::locale, 2396
- ctype_base.h, 3732
- ctype_inline.h, 3732
- cuchar, 3512
- cur
 - std::basic_fstream< _CharT, _Traits >, 1188
 - std::basic_ifstream< _CharT, _Traits >, 1231
 - std::basic_ios< _CharT, _Traits >, 1255
 - std::basic_iostream< _CharT, _Traits >, 1302
 - std::basic_istream< _CharT, _Traits >, 1342
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1382
 - std::basic_ofstream< _CharT, _Traits >, 1416
 - std::basic_ostream< _CharT, _Traits >, 1447
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1480
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1765
 - std::ios_base, 2306
- curr_symbol
 - std::moneypunct< _CharT, _Intl >, 2491
 - std::moneypunct_byname< _CharT, _Intl >, 2499
- current_exception
 - Exceptions, 135
- cv_status
 - Condition Variables, 118
- cwchar, 3512, 3513
- cwctype, 3513
- cxxabi.h, 3736
 - __cxa_demangle, 3737
- cxxabi_forced.h, 3326
- cxxabi_init_exception.h, 3326
- cxxabi_tweaks.h, 3733
- cyl_bessel_i
 - Mathematical Special Functions, 204
 - TR1 Mathematical Special Functions, 254
- cyl_bessel_if
 - Mathematical Special Functions, 205
- cyl_bessel_il
 - Mathematical Special Functions, 205
- cyl_bessel_j
 - Mathematical Special Functions, 205
 - TR1 Mathematical Special Functions, 254
- cyl_bessel_jf
 - Mathematical Special Functions, 206
- cyl_bessel_jl
 - Mathematical Special Functions, 206
- cyl_bessel_k
 - Mathematical Special Functions, 206
 - TR1 Mathematical Special Functions, 254
- cyl_bessel_kf
 - Mathematical Special Functions, 207
- cyl_bessel_kl
 - Mathematical Special Functions, 207
- cyl_neumann
 - Mathematical Special Functions, 208
 - TR1 Mathematical Special Functions, 254
- cyl_neumannf
 - Mathematical Special Functions, 208
- cyl_neumannl
 - Mathematical Special Functions, 208
- data
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 854
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1534
 - std, 602, 603
 - std::basic_string< _CharT, _Traits, _Alloc >, 1630, 1631
 - std::vector< _Tp, _Alloc >, 3244
- Data parallel extensions, 301
- Data Structure Type, 151
- date_order
 - std::time_get< _CharT, _Intl >, 3019
 - std::time_get_byname< _CharT, _Intl >, 3031
- days
 - Time, 375
- deallocate
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 761, 762
 - std::allocator_traits< _Alloc >, 1066
 - std::allocator_traits< allocator< _Tp > >, 1072
 - std::allocator_traits< allocator< void > >, 1076
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1081
- debug.h, 3514

- debug_allocator.h, [3546](#)
- debug_fn_imps.hpp, [3555](#), [3556](#)
- debug_map_base.hpp, [3573](#)
- debug_no_store_hash_fn_imps.hpp, [3569](#)
- debug_store_hash_fn_imps.hpp, [3570](#)
- debugging, [3529](#)
- dec
 - std, [603](#)
 - std::basic_fstream< _CharT, _Traits >, [1188](#)
 - std::basic_ifstream< _CharT, _Traits >, [1231](#)
 - std::basic_ios< _CharT, _Traits >, [1255](#)
 - std::basic_iostream< _CharT, _Traits >, [1302](#)
 - std::basic_istream< _CharT, _Traits >, [1342](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1382](#)
 - std::basic_ofstream< _CharT, _Traits >, [1416](#)
 - std::basic_ostream< _CharT, _Traits >, [1447](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1481](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1765](#)
 - std::ios_base, [2306](#)
- decay_t
 - Metaprogramming, [360](#)
- decimal, [3530](#)
- Decimal Floating-Point Arithmetic, [191](#)
- decimal128
 - std::decimal::decimal128, [2050](#)
- decimal32
 - std::decimal::decimal32, [2052](#)
- decimal32_to_long_long
 - std::decimal, [703](#)
- decimal64
 - std::decimal::decimal64, [2053](#)
- decimal_point
 - std::moneypunct< _CharT, _Intl >, [2491](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2499](#)
 - std::numpunct< _CharT >, [2686](#)
 - std::numpunct_byname< _CharT >, [2690](#)
- declval
 - Utilities, [307](#)
- default_delete
 - std::default_delete< _Tp >, [2055](#)
 - std::default_delete< _Tp[]>, [2056](#)
- default_error_condition
 - std::error_category, [2148](#)
 - std::error_code, [2150](#)
- default_sentinel
 - std, [657](#)
- defaultfloat
 - std, [603](#)
- defer_lock
 - Mutexes, [124](#)
- denorm_absent
 - std, [592](#)
- denorm_indeterminate
 - std, [592](#)
- denorm_min
 - std::numeric_limits< _Tp >, [2607](#)
- denorm_present
 - std, [592](#)
- densities
 - std::piecewise_constant_distribution< _RealType >, [2762](#)
 - std::piecewise_linear_distribution< _RealType >, [2766](#)
- Deprecated List, [1](#)
- deque, [3539](#), [3540](#)
 - std::deque< _Tp, _Alloc >, [2068–2070](#)
- deque.tcc, [3327](#)
- destroy
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [762](#)
 - std::allocator_traits< _Alloc >, [1067](#)
 - std::allocator_traits< allocator< _Tp > >, [1072](#)
 - std::allocator_traits< allocator< void > >, [1076](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1081](#)
- destroying_delete
 - std, [657](#)
- detected_or
 - Detection idiom, [291](#)
- detected_or_t
 - Detection idiom, [291](#)
- detected_t
 - Detection idiom, [291](#)
- Detection idiom, [290](#)
 - __cpp_lib_experimental_detect, [291](#)
 - detected_or, [291](#)
 - detected_or_t, [291](#)
 - detected_t, [291](#)
 - is_detected, [291](#)
 - is_detected_convertible, [292](#)
 - is_detected_convertible_v, [292](#)
 - is_detected_exact, [292](#)
 - is_detected_exact_v, [292](#)
 - is_detected_v, [292](#)
 - void_t, [292](#)
- Diagnostics, [129](#)
 - generic_category, [130](#)
 - make_error_code, [130](#)
 - make_error_condition, [130](#)
 - operator<=, [131](#)
 - operator<=>, [131](#)
 - operator==, [132](#)
 - system_category, [133](#)
- difference_type
 - __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, [1775](#)

- __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_iterator, Iterator, _Alloc >, 1779
- __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1789
- __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1793
- __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2345
- __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2349
- std::allocator_traits< _Alloc >, 1064
- std::allocator_traits< allocator< _Tp > >, 1069
- std::allocator_traits< allocator< void > >, 1074
- std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
- std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2331
- std::istreambuf_iterator< _CharT, _Traits >, 2334
- std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2339
- std::ostream_iterator< _Tp, _CharT, _Traits >, 2698
- std::ostreambuf_iterator< _CharT, _Traits >, 2702
- std::pointer_traits< _Ptr >, 2773
- std::pointer_traits< _Tp * >, 2774
- std::raw_storage_iterator< _OutputIterator, _Tp >, 2814
- std::set< _Key, _Compare, _Alloc >, 2891
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3125
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3154
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- digits
 - std::__numeric_limits_base, 815
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621
 - std::numeric_limits< char32_t >, 2625
 - std::numeric_limits< double >, 2629
 - std::numeric_limits< float >, 2633
 - std::numeric_limits< int >, 2637
 - std::numeric_limits< long >, 2641
 - std::numeric_limits< long double >, 2645
 - std::numeric_limits< long long >, 2649
 - std::numeric_limits< short >, 2653
 - std::numeric_limits< signed char >, 2657
 - std::numeric_limits< unsigned char >, 2661
 - std::numeric_limits< unsigned int >, 2665
 - std::numeric_limits< unsigned long >, 2669
 - std::numeric_limits< unsigned long long >, 2673
- std::numeric_limits< unsigned short >, 2677
- std::numeric_limits< wchar_t >, 2681
- digits10
 - std::__numeric_limits_base, 815
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621
 - std::numeric_limits< char32_t >, 2625
 - std::numeric_limits< double >, 2629
 - std::numeric_limits< float >, 2633
 - std::numeric_limits< int >, 2637
 - std::numeric_limits< long >, 2641
 - std::numeric_limits< long double >, 2645
 - std::numeric_limits< long long >, 2649
 - std::numeric_limits< short >, 2653
 - std::numeric_limits< signed char >, 2657
 - std::numeric_limits< unsigned char >, 2661
 - std::numeric_limits< unsigned int >, 2665
 - std::numeric_limits< unsigned long >, 2669
 - std::numeric_limits< unsigned long long >, 2673
 - std::numeric_limits< unsigned short >, 2677
 - std::numeric_limits< wchar_t >, 2681
- direct_mask_range_hashing_imp.hpp, 3574
- direct_mod_range_hashing_imp.hpp, 3574
- directory_options
 - Filesystem, 167
 - Filesystem TS, 286
- discard
 - std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2093
 - std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2281
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2359
 - std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2464
 - std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2758
 - std::shuffle_order_engine< _RandomNumberEngine, __k >, 2926
 - std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2997
- discard_block_engine
 - std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2092, 2093
- distance
 - SGL, 159
 - std, 603
- do_allocate
 - std::pmr::monotonic_buffer_resource, 2506
 - std::pmr::unsynchronized_pool_resource, 3223
- do_always_noconv

- std::codecvt< _InternT, _ExternT, _StateT >, [1854](#)
- std::codecvt< _InternT, _ExternT, encoding_state >, [1860](#)
- std::codecvt< char, char, mbstate_t >, [1867](#)
- std::codecvt< char16_t, char, mbstate_t >, [1875](#)
- std::codecvt< char32_t, char, mbstate_t >, [1882](#)
- std::codecvt< wchar_t, char, mbstate_t >, [1890](#)
- std::codecvt_byname< _InternT, _ExternT, _StateT >, [1898](#)
- do_compare
 - std::__cxx11::collate< _CharT >, [1905](#)
- do_curr_symbol
 - std::moneypunct< _CharT, _Intl >, [2492](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2499](#)
- do_date_order
 - std::time_get< _CharT, _Inlter >, [3020](#)
 - std::time_get_byname< _CharT, _Inlter >, [3031](#)
- do_deallocate
 - std::pmr::monotonic_buffer_resource, [2506](#)
 - std::pmr::unsynchronized_pool_resource, [3223](#)
- do_decimal_point
 - std::moneypunct< _CharT, _Intl >, [2492](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2499](#)
 - std::numpunct< _CharT >, [2686](#)
 - std::numpunct_byname< _CharT >, [2690](#)
- do_encoding
 - std::codecvt< _InternT, _ExternT, _StateT >, [1854](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1860](#)
 - std::codecvt< char, char, mbstate_t >, [1867](#)
 - std::codecvt< char16_t, char, mbstate_t >, [1875](#)
 - std::codecvt< char32_t, char, mbstate_t >, [1882](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1890](#)
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, [1898](#)
- do_falsename
 - std::numpunct< _CharT >, [2686](#)
 - std::numpunct_byname< _CharT >, [2690](#)
- do_frac_digits
 - std::moneypunct< _CharT, _Intl >, [2492](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2500](#)
- do_get
 - std::messages< _CharT >, [2469](#)
 - std::messages_byname< _CharT >, [2472](#)
 - std::money_get< _CharT, _Inlter >, [2481](#)
 - std::num_get< _CharT, _Inlter >, [2578–2585](#)
 - std::time_get< _CharT, _Inlter >, [3020](#)
 - std::time_get_byname< _CharT, _Inlter >, [3031](#)
- do_get_date
 - std::time_get< _CharT, _Inlter >, [3021](#)
 - std::time_get_byname< _CharT, _Inlter >, [3032](#)
- do_get_monthname
 - std::time_get< _CharT, _Inlter >, [3021](#)
 - std::time_get_byname< _CharT, _Inlter >, [3032](#)
- do_get_time
 - std::time_get< _CharT, _Inlter >, [3022](#)
 - std::time_get_byname< _CharT, _Inlter >, [3033](#)
- do_get_weekday
 - std::time_get< _CharT, _Inlter >, [3023](#)
 - std::time_get_byname< _CharT, _Inlter >, [3034](#)
- do_get_year
 - std::time_get< _CharT, _Inlter >, [3023](#)
 - std::time_get_byname< _CharT, _Inlter >, [3034](#)
- do_grouping
 - std::moneypunct< _CharT, _Intl >, [2492](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2500](#)
 - std::numpunct< _CharT >, [2686](#)
 - std::numpunct_byname< _CharT >, [2691](#)
- do_hash
 - std::__cxx11::collate< _CharT >, [1905](#)
- do_in
 - std::codecvt< _InternT, _ExternT, _StateT >, [1855](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1860](#)
 - std::codecvt< char, char, mbstate_t >, [1868](#)
 - std::codecvt< char16_t, char, mbstate_t >, [1875](#)
 - std::codecvt< char32_t, char, mbstate_t >, [1882](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1890](#)
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, [1898](#)
- do_is
 - std::__ctype_abstract_base< _CharT >, [777](#)
 - std::ctype< _CharT >, [1954](#)
 - std::ctype< char >, [1969, 1970](#)
 - std::ctype< wchar_t >, [1999](#)
 - std::ctype_byname< _CharT >, [2013](#)
 - std::ctype_byname< char >, [2027, 2028](#)
- do_is_equal
 - std::pmr::monotonic_buffer_resource, [2506](#)
 - std::pmr::unsynchronized_pool_resource, [3223](#)
- do_length
 - std::codecvt< _InternT, _ExternT, _StateT >, [1855](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1860](#)
 - std::codecvt< char, char, mbstate_t >, [1868](#)
 - std::codecvt< char16_t, char, mbstate_t >, [1875](#)
 - std::codecvt< char32_t, char, mbstate_t >, [1883](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1890](#)
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, [1899](#)
- do_max_length
 - std::codecvt< _InternT, _ExternT, _StateT >, [1855](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1860](#)
 - std::codecvt< char, char, mbstate_t >, [1868](#)
 - std::codecvt< char16_t, char, mbstate_t >, [1875](#)
 - std::codecvt< char32_t, char, mbstate_t >, [1883](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1890](#)

- std::codecvt_byname< _InternT, _ExternT, _StateT >, 1899
- do_narrow
 - std::__ctype_abstract_base< _CharT >, 777, 778
 - std::ctype< _CharT >, 1954, 1955
 - std::ctype< char >, 1971, 1972
 - std::ctype< wchar_t >, 1999, 2000
 - std::ctype_byname< _CharT >, 2013, 2014
 - std::ctype_byname< char >, 2029, 2030
- do_neg_format
 - std::moneypunct< _CharT, _Intl >, 2493
 - std::moneypunct_byname< _CharT, _Intl >, 2500
- do_negative_sign
 - std::moneypunct< _CharT, _Intl >, 2493
 - std::moneypunct_byname< _CharT, _Intl >, 2501
- do_out
 - std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 768
 - std::codecvt< _InternT, _ExternT, _StateT >, 1855
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1861
 - std::codecvt< char, char, mbstate_t >, 1868
 - std::codecvt< char16_t, char, mbstate_t >, 1875
 - std::codecvt< char32_t, char, mbstate_t >, 1883
 - std::codecvt< wchar_t, char, mbstate_t >, 1890
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, 1899
- do_pos_format
 - std::moneypunct< _CharT, _Intl >, 2493
 - std::moneypunct_byname< _CharT, _Intl >, 2501
- do_positive_sign
 - std::moneypunct< _CharT, _Intl >, 2494
 - std::moneypunct_byname< _CharT, _Intl >, 2501
- do_put
 - std::money_put< _CharT, _Outlter >, 2485, 2486
 - std::num_put< _CharT, _Outlter >, 2596–2599
 - std::time_put< _CharT, _Outlter >, 3043
 - std::time_put_byname< _CharT, _Outlter >, 3046
- do_scan_is
 - std::__ctype_abstract_base< _CharT >, 779
 - std::ctype< _CharT >, 1956
 - std::ctype< char >, 1973
 - std::ctype< wchar_t >, 2000
 - std::ctype_byname< _CharT >, 2015
 - std::ctype_byname< char >, 2031
- do_scan_not
 - std::__ctype_abstract_base< _CharT >, 779
 - std::ctype< _CharT >, 1956
 - std::ctype< char >, 1974
 - std::ctype< wchar_t >, 2001
 - std::ctype_byname< _CharT >, 2015
 - std::ctype_byname< char >, 2031, 2032
- do_thousands_sep
 - std::moneypunct< _CharT, _Intl >, 2494
- std::moneypunct_byname< _CharT, _Intl >, 2502
- std::numput< _CharT >, 2687
- std::numput_byname< _CharT >, 2691
- do_tolower
 - std::__ctype_abstract_base< _CharT >, 780
 - std::ctype< _CharT >, 1957
 - std::ctype< char >, 1975, 1976
 - std::ctype< wchar_t >, 2001, 2002
 - std::ctype_byname< _CharT >, 2016
 - std::ctype_byname< char >, 2032–2034
- do_toupper
 - std::__ctype_abstract_base< _CharT >, 781
 - std::ctype< _CharT >, 1957, 1958
 - std::ctype< char >, 1977, 1978
 - std::ctype< wchar_t >, 2002, 2003
 - std::ctype_byname< _CharT >, 2016, 2017
 - std::ctype_byname< char >, 2034, 2035
- do_transform
 - std::__cxx11::collate< _CharT >, 1906
- do_truename
 - std::numput< _CharT >, 2687
 - std::numput_byname< _CharT >, 2691
- do_unshift
 - std::codecvt< _InternT, _ExternT, _StateT >, 1855
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1861
 - std::codecvt< char, char, mbstate_t >, 1868
 - std::codecvt< char16_t, char, mbstate_t >, 1876
 - std::codecvt< char32_t, char, mbstate_t >, 1883
 - std::codecvt< wchar_t, char, mbstate_t >, 1891
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, 1899
- do_widen
 - std::__ctype_abstract_base< _CharT >, 781, 782
 - std::ctype< _CharT >, 1958, 1959
 - std::ctype< char >, 1978, 1979
 - std::ctype< wchar_t >, 2003
 - std::ctype_byname< _CharT >, 2017, 2018
 - std::ctype_byname< char >, 2036, 2037
- duration_cast
 - Time, 378
- Dynamic Bitset., 139
 - operator!=, 140
 - operator<<, 141
 - operator<=, 141
 - operator>, 141
 - operator>>, 142
 - operator>=, 142
 - operator-, 141
 - operator&, 140
 - operator^, 142
 - operator|, 143
- dynamic_bitset, 3705

- `std::tr2::dynamic_bitset< _WordT, _Alloc >`, [2107–2109](#)
- `dynamic_bitset.tcc`, [3706](#)
- `dynamic_pointer_cast`
 - Pointer Abstractions, [348](#)
 - `std`, [604](#)
- `e_pos`
 - `__gnu_pbds::sample_trie_access_traits`, [2874](#)
 - `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >`, [3076](#)
- `e_type`
 - `__gnu_pbds::sample_trie_access_traits`, [2874](#)
 - `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >`, [3075](#)
- `eback`
 - `__gnu_cxx::enc_filebuf< _CharT >`, [2123](#)
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [2949](#)
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [2969](#)
 - `std::basic_filebuf< _CharT, _Traits >`, [1124](#)
 - `std::basic_streambuf< _CharT, _Traits >`, [1498](#)
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [1705](#)
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3262](#)
- ECMAScript
 - `std::regex_constants`, [728](#)
- `egptr`
 - `__gnu_cxx::enc_filebuf< _CharT >`, [2123](#)
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [2949](#)
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [2969](#)
 - `std::basic_filebuf< _CharT, _Traits >`, [1124](#)
 - `std::basic_streambuf< _CharT, _Traits >`, [1499](#)
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [1705](#)
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3262](#)
- `egrep`
 - `std::regex_constants`, [728](#)
- `element_type`
 - `std::auto_ptr< _Tp >`, [1101](#)
 - `std::pointer_traits< _Ptr >`, [2773](#)
 - `std::pointer_traits< _Tp * >`, [2774](#)
 - `std::shared_ptr< _Tp >`, [2916](#)
- `ellint_1`
 - Mathematical Special Functions, [209](#)
 - TR1 Mathematical Special Functions, [254](#)
- `ellint_1f`
 - Mathematical Special Functions, [209](#)
- `ellint_1l`
 - Mathematical Special Functions, [209](#)
- `ellint_2`
 - Mathematical Special Functions, [210](#)
 - TR1 Mathematical Special Functions, [255](#)
- `ellint_2f`
 - Mathematical Special Functions, [210](#)
- `ellint_2l`
 - Mathematical Special Functions, [211](#)
- `ellint_3`
 - Mathematical Special Functions, [211](#)
 - TR1 Mathematical Special Functions, [255](#)
- `ellint_3f`
 - Mathematical Special Functions, [212](#)
- `ellint_3l`
 - Mathematical Special Functions, [212](#)
- `emplace`
 - `std::deque< _Tp, _Alloc >`, [2077](#)
 - `std::list< _Tp, _Alloc >`, [2374](#)
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2428](#)
 - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [2522](#)
 - `std::multiset< _Key, _Compare, _Alloc >`, [2546](#)
 - `std::set< _Key, _Compare, _Alloc >`, [2897](#)
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3133](#)
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3161](#)
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3186](#)
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3210](#)
 - `std::vector< _Tp, _Alloc >`, [3244](#)
- `emplace_after`
 - `std::forward_list< _Tp, _Alloc >`, [2186](#)
- `emplace_front`
 - `std::forward_list< _Tp, _Alloc >`, [2186](#)
- `emplace_hint`
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2429](#)
 - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [2523](#)
 - `std::multiset< _Key, _Compare, _Alloc >`, [2546](#)
 - `std::set< _Key, _Compare, _Alloc >`, [2898](#)
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3134](#)
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3161](#)
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3186](#)
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3210](#)
- `empty`
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [854](#)
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [1534](#)
 - `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`, [1839](#)

```

__gnu_pbds::detail::gp_ht_map< Key, Mapped,
    Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
    Comb_Probe_Fn, Probe_Fn, Resize_Policy
    >, 2232
std, 604
std::basic_string< _CharT, _Traits, _Alloc >, 1631
std::deque< _Tp, _Alloc >, 2077
std::experimental::fundamentals_v1::any, 1085
std::forward_list< _Tp, _Alloc >, 2186
std::list< _Tp, _Alloc >, 2375
std::map< _Key, _Tp, _Compare, _Alloc >, 2429
std::match_results< _Bi_iter, _Alloc >, 2449
std::multimap< _Key, _Tp, _Compare, _Alloc >,
    2523
std::multiset< _Key, _Compare, _Alloc >, 2547
std::priority_queue< _Tp, _Sequence, _Compare >,
    2787
std::queue< _Tp, _Sequence >, 2795
std::set< _Key, _Compare, _Alloc >, 2898
std::stack< _Tp, _Sequence >, 2943
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2110
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Al-
    loc >, 3134
std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
    _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Al-
    loc >, 3187
std::unordered_set< _Value, _Hash, _Pred, _Alloc
    >, 3211
std::vector< _Tp, _Alloc >, 3244
enable_if_t
    Metaprogramming, 360
enable_special_members.h, 3328
enable_view
    ranges_base.h, 3384
enc_filebuf.h, 3547
end
__gnu_cxx::Temporary_buffer< _ForwardIterator,
    _Tp >, 1048
__gnu_cxx::versa_string< _CharT, _Traits, _Alloc,
    _Base >, 854
__gnu_cxx::temporary_buffer< _ForwardIterator,
    _Tp >, 3005
__gnu_parallel::PseudoSequence< _Tp, _Differ-
    enceTp >, 978
__gnu_pbds::detail::trie_policy_base< Node_Cltr,
    Node_Itr, _ATraits, _Alloc >, 3071
__gnu_pbds::sample_trie_access_traits, 2875
__gnu_pbds::trie_string_access_traits< String,
    Min_E_Val, Max_E_Val, Reverse, _Alloc >,
    3076
Filesystem, 167, 168
std, 605
std::_Temporary_buffer< _ForwardIterator, _Tp >,
    1050
std::basic_fstream< _CharT, _Traits >, 1188
std::basic_ifstream< _CharT, _Traits >, 1231
std::basic_ios< _CharT, _Traits >, 1256
std::basic_iostream< _CharT, _Traits >, 1303
std::basic_istream< _CharT, _Traits >, 1342
std::basic_istreamstream< _CharT, _Traits, _Alloc >,
    1383
std::basic_ofstream< _CharT, _Traits >, 1416
std::basic_ostream< _CharT, _Traits >, 1448
std::basic_ostreamstream< _CharT, _Traits, _Alloc >,
    1481
std::basic_string< _CharT, _Traits, _Alloc >, 1631
std::basic_stringstream< _CharT, _Traits, _Alloc >,
    1765
std::deque< _Tp, _Alloc >, 2077, 2078
std::forward_list< _Tp, _Alloc >, 2186, 2187
std::ios_base, 2306
std::list< _Tp, _Alloc >, 2375
std::map< _Key, _Tp, _Compare, _Alloc >, 2430
std::match_results< _Bi_iter, _Alloc >, 2449
std::multimap< _Key, _Tp, _Compare, _Alloc >,
    2523
std::multiset< _Key, _Compare, _Alloc >, 2547
std::set< _Key, _Compare, _Alloc >, 2898
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Al-
    loc >, 3135
std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
    _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Al-
    loc >, 3187, 3188
std::unordered_set< _Value, _Hash, _Pred, _Alloc
    >, 3211, 3212
std::valarray< _Tp >, 3227
std::vector< _Tp, _Alloc >, 3244
endl
std, 606
ends
std, 606
entry_cmp.hpp, 3557
entry_list_fn_imps.hpp, 3570
entry_metadata_base.hpp, 3579
entry_pred.hpp, 3557
eof
std::basic_fstream< _CharT, _Traits >, 1152
std::basic_ifstream< _CharT, _Traits >, 1203
std::basic_ios< _CharT, _Traits >, 1244
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1315
std::basic_istreamstream< _CharT, _Traits, _Alloc >,
    1356
std::basic_ofstream< _CharT, _Traits >, 1395
std::basic_ostream< _CharT, _Traits >, 1428

```

- std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1461
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1731
- eofbit
 - std::basic_fstream< _CharT, _Traits >, 1188
 - std::basic_ifstream< _CharT, _Traits >, 1231
 - std::basic_ios< _CharT, _Traits >, 1256
 - std::basic_iostream< _CharT, _Traits >, 1303
 - std::basic_istream< _CharT, _Traits >, 1342
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1383
 - std::basic_ofstream< _CharT, _Traits >, 1416
 - std::basic_ostream< _CharT, _Traits >, 1448
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1481
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1765
 - std::ios_base, 2306
- ep_ptr
 - __gnu_cxx::enc_filebuf< _CharT >, 2124
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2949
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2969
 - std::basic_filebuf< _CharT, _Traits >, 1125
 - std::basic_streambuf< _CharT, _Traits >, 1499
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1706
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3262
- epsilon
 - std::numeric_limits< _Tp >, 2607
- eq_by_less.hpp, 3573
- equal
 - Non-Mutating, 48, 49
 - std::istreambuf_iterator< _CharT, _Traits >, 2335
- equal_range
 - Binary Search, 87, 88
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2430
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2524
 - std::multiset< _Key, _Compare, _Alloc >, 2547
 - std::set< _Key, _Compare, _Alloc >, 2899
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3135, 3136
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3163
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3188
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3212
- equally_split.h, 3665
- equals
 - std::tr2::bool_set, 1822
- equivalent
 - std::error_category, 2148
- erase
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 855
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1534–1536
 - std::basic_string< _CharT, _Traits, _Alloc >, 1632–1634
 - std::deque< _Tp, _Alloc >, 2078
 - std::list< _Tp, _Alloc >, 2375, 2376
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2431, 2432
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2524–2526
 - std::multiset< _Key, _Compare, _Alloc >, 2548, 2549
 - std::set< _Key, _Compare, _Alloc >, 2899, 2900
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3136–3138
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3163–3165
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3189, 3190
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3213, 3214
 - std::vector< _Tp, _Alloc >, 3245
- erase_after
 - std::chrono::tzdb_list, 3094
 - std::forward_list< _Tp, _Alloc >, 2187
- erase_can_throw
 - __gnu_pbds::container_traits< Cntr >, 1941
- erase_fn_imps.hpp, 3557, 3558
- erase_if.h, 3329
- erase_no_store_hash_fn_imps.hpp, 3570
- erase_store_hash_fn_imps.hpp, 3570
- error_backref
 - std::regex_constants, 723
- error_badbrace
 - std::regex_constants, 723
- error_badrepeat
 - std::regex_constants, 724
- error_brace
 - std::regex_constants, 724
- error_brack
 - std::regex_constants, 724
- error_code
 - std::error_code, 2150
- error_collate
 - std::regex_constants, 724
- error_complexity
 - std::regex_constants, 724
- error_condition
 - std::error_condition, 2151
- error_constants.h, 3733
- error_ctype
 - std::regex_constants, 724

- error_escape
 - std::regex_constants, 724
- error_paren
 - std::regex_constants, 724
- error_range
 - std::regex_constants, 724
- error_space
 - std::regex_constants, 725
- error_stack
 - std::regex_constants, 725
- error_type
 - std::regex_constants, 723
- event
 - std::basic_fstream< _CharT, _Traits >, 1150
 - std::basic_ifstream< _CharT, _Traits >, 1201
 - std::basic_ios< _CharT, _Traits >, 1242
 - std::basic_iostream< _CharT, _Traits >, 1267
 - std::basic_istream< _CharT, _Traits >, 1313
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1353
 - std::basic_ofstream< _CharT, _Traits >, 1393
 - std::basic_ostream< _CharT, _Traits >, 1426
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1459
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1728
 - std::ios_base, 2299
- event_callback
 - std::basic_fstream< _CharT, _Traits >, 1148
 - std::basic_ifstream< _CharT, _Traits >, 1199
 - std::basic_ios< _CharT, _Traits >, 1240
 - std::basic_iostream< _CharT, _Traits >, 1265
 - std::basic_istream< _CharT, _Traits >, 1311
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1351
 - std::basic_ofstream< _CharT, _Traits >, 1391
 - std::basic_ostream< _CharT, _Traits >, 1424
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1457
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1727
 - std::ios_base, 2298
- exception, 3738
- exception.h, 3329
- exception.hpp, 3591
- exception_defines.h, 3329
- exception_ptr.h, 3329
 - rethrow_exception, 3330
- Exceptions, 133, 149
 - __verbose_terminate_handler, 135
 - current_exception, 135
 - get_terminate, 135
 - make_exception_ptr, 136
 - rethrow_exception, 136
 - rethrow_if_nested, 136
 - set_terminate, 136
 - terminate, 137
 - terminate_handler, 135
 - throw_with_nested, 137
 - uncaught_exception, 137
- exceptions
 - std::basic_fstream< _CharT, _Traits >, 1153
 - std::basic_ifstream< _CharT, _Traits >, 1204
 - std::basic_ios< _CharT, _Traits >, 1244
 - std::basic_iostream< _CharT, _Traits >, 1268, 1269
 - std::basic_istream< _CharT, _Traits >, 1315
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1356
 - std::basic_ofstream< _CharT, _Traits >, 1396
 - std::basic_ostream< _CharT, _Traits >, 1428
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1461
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1731
- exclusive_scan
 - Generalized Numeric operations, 8, 9
- exp
 - Complex Numbers, 186
 - math.h, 3635
- expected, 3540
- expint
 - Mathematical Special Functions, 212
 - TR1 Mathematical Special Functions, 255
- expintf
 - Mathematical Special Functions, 213
- expintl
 - Mathematical Special Functions, 213
- explicit
 - __gnu_cxx::pair< _T1, _T2 >, 2722
 - __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 925–927
 - std::complex< _Tp >, 1913
 - std::pair< _T1, _T2 >, 2727, 2728
- exponential_distribution
 - std::exponential_distribution< _RealType >, 2158
- extc++.h, 3734
- extended
 - std::regex_constants, 728
- Extensions, 138
- external_load_access
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1827
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2263
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2273
- extptr_allocator.h, 3547

- fabs
 - Complex Numbers, [186](#)
 - math.h, [3636](#)
 - std, [606](#)
- facet
 - std::locale::facet, [2165](#)
- fail
 - std::basic_fstream< _CharT, _Traits >, [1153](#)
 - std::basic_ifstream< _CharT, _Traits >, [1204](#)
 - std::basic_ios< _CharT, _Traits >, [1245](#)
 - std::basic_iostream< _CharT, _Traits >, [1269](#)
 - std::basic_istream< _CharT, _Traits >, [1316](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1357](#)
 - std::basic_ofstream< _CharT, _Traits >, [1396](#)
 - std::basic_ostream< _CharT, _Traits >, [1429](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1462](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1732](#)
- failbit
 - std::basic_fstream< _CharT, _Traits >, [1188](#)
 - std::basic_ifstream< _CharT, _Traits >, [1231](#)
 - std::basic_ios< _CharT, _Traits >, [1256](#)
 - std::basic_iostream< _CharT, _Traits >, [1303](#)
 - std::basic_istream< _CharT, _Traits >, [1342](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1383](#)
 - std::basic_ofstream< _CharT, _Traits >, [1416](#)
 - std::basic_ostream< _CharT, _Traits >, [1448](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1481](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1765](#)
 - std::ios_base, [2306](#)
- failed
 - std::ostreambuf_iterator< _CharT, _Traits >, [2703](#)
- false_type
 - Metaprogramming, [360](#)
- falsename
 - std::numpunct< _CharT >, [2687](#)
 - std::numpunct_byname< _CharT >, [2691](#)
- fd
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2950](#)
- features.h, [3665](#)
 - _GLIBCXX_BAL_QUICKSORT, [3665](#)
 - _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, [3665](#)
 - _GLIBCXX_FIND_EQUAL_SPLIT, [3666](#)
 - _GLIBCXX_FIND_GROWING_BLOCKS, [3666](#)
 - _GLIBCXX_MERGESORT, [3666](#)
 - _GLIBCXX_QUICKSORT, [3666](#)
 - _GLIBCXX_TREE_DYNAMIC_BALANCING, [3666](#)
 - _GLIBCXX_TREE_FULL_COPY, [3666](#)
- __GLIBCXX_TREE_INITIAL_SPLITTING, [3667](#)
- fenv.h, [3617](#)
- file
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2950](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2970](#)
- file_time_type
 - Filesystem, [167](#)
 - Filesystem TS, [285](#)
- file_type
 - Filesystem, [167](#)
 - Filesystem TS, [286](#)
- filebuf
 - I/O, [170](#)
- Filesystem, [162](#)
 - begin, [167](#)
 - copy_options, [167](#)
 - directory_options, [167](#)
 - end, [167](#), [168](#)
 - file_time_type, [167](#)
 - file_type, [167](#)
 - perm_options, [167](#)
 - perms, [167](#)
 - u8path, [168](#)
- filesystem, [3617](#)
- Filesystem TS, [281](#)
 - copy_options, [286](#)
 - directory_options, [286](#)
 - file_time_type, [285](#)
 - file_type, [286](#)
 - perms, [286](#)
- fill
 - Mutating, [25](#)
 - std::basic_fstream< _CharT, _Traits >, [1154](#)
 - std::basic_ifstream< _CharT, _Traits >, [1205](#)
 - std::basic_ios< _CharT, _Traits >, [1245](#)
 - std::basic_iostream< _CharT, _Traits >, [1269](#), [1270](#)
 - std::basic_istream< _CharT, _Traits >, [1316](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1357](#)
 - std::basic_ofstream< _CharT, _Traits >, [1397](#)
 - std::basic_ostream< _CharT, _Traits >, [1429](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1462](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1732](#)
- fill_minimal_n
 - __gnu_parallel::Settings, [1040](#)
- fill_n
 - Mutating, [26](#)
- find
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [856](#), [857](#)

- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1536–1541
- Non-Mutating, 50
- `std::basic_string< _CharT, _Traits, _Alloc >`, 1634–1637
- `std::map< _Key, _Tp, _Compare, _Alloc >`, 2433
- `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2526
- `std::multiset< _Key, _Compare, _Alloc >`, 2549
- `std::set< _Key, _Compare, _Alloc >`, 2901
- `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3138, 3139
- `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3165, 3166
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3190, 3191
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3214, 3215
- `find.h`, 3667
- `find_by_order`
 - `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`, 3055
 - `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3069
- `find_end`
 - Non-Mutating, 50, 51
- `find_first`
 - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2110
- `find_first_not_of`
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 858, 859
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1541–1545
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1638–1641
- `find_first_of`
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 860, 861
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1546–1550
 - Non-Mutating, 52
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1641–1644
- `find_fn_imps.hpp`, 3559, 3560
- `find_if`
 - Non-Mutating, 53
- `find_if_not`
 - Non-Mutating, 53
- `find_increasing_factor`
 - `__gnu_parallel::Settings`, 1040
- `find_initial_block_size`
 - `__gnu_parallel::Settings`, 1040
- `find_last_not_of`
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 861–863
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1551–1555
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1644–1647
- `find_last_of`
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 863–865
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1556–1560
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1648–1651
- `find_maximum_block_size`
 - `__gnu_parallel::Settings`, 1040
- `find_next`
 - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2110
- `find_no_store_hash_fn_imps.hpp`, 3573
- `find_scale_factor`
 - `__gnu_parallel::Settings`, 1040
- `find_selectors.h`, 3667
- `find_sequential_search_size`
 - `__gnu_parallel::Settings`, 1040
- `find_store_hash_fn_imps.hpp`, 3570
- `first`
 - `__gnu_cxx::pair< _T1, _T2 >`, 2724
 - `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 928
 - `std::pair< _T1, _T2 >`, 2729
- `first_argument_type`
 - `__gnu_cxx::equal_to< _Tp >`, 2144
 - `__gnu_cxx::project1st< _Arg1, _Arg2 >`, 2789
 - `__gnu_cxx::project2nd< _Arg1, _Arg2 >`, 2789
 - `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`, 905
 - `__gnu_parallel::_EqualTo< _T1, _T2 >`, 906
 - `__gnu_parallel::_Less< _T1, _T2 >`, 932
 - `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >`, 933
 - `__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >`, 935
 - `__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >`, 961
 - `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >`, 973
 - `std::binary_function< _Arg1, _Arg2, _Result >`, 1786
 - `std::binary_negate< _Predicate >`, 1796
 - `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 1923
 - `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 1924
 - `std::divides< _Tp >`, 2100
 - `std::divides< void >`, 2101
 - `std::equal_to< _Tp >`, 2145
 - `std::equal_to< void >`, 2147
 - `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >`, 2715

- std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2717
- std::greater< _Tp >, 2235
- std::greater< void >, 2237
- std::greater_equal< _Tp >, 2238
- std::greater_equal< void >, 2239
- std::less< _Tp >, 2352
- std::less< void >, 2354
- std::less_equal< _Tp >, 2355
- std::less_equal< void >, 2356
- std::logical_and< _Tp >, 2400
- std::logical_and< void >, 2401
- std::logical_or< _Tp >, 2404
- std::logical_or< void >, 2405
- std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2457
- std::mem_fun1_t< _Ret, _Tp, _Arg >, 2458
- std::minus< _Tp >, 2473
- std::minus< void >, 2474
- std::modulus< _Tp >, 2476
- std::modulus< void >, 2477
- std::multiplies< _Tp >, 2535
- std::multiplies< void >, 2536
- std::not_equal_to< _Tp >, 2572
- std::not_equal_to< void >, 2574
- std::owner_less< shared_ptr< _Tp > >, 2716
- std::owner_less< void >, 2717
- std::owner_less< weak_ptr< _Tp > >, 2718
- std::plus< _Tp >, 2768
- std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2771
- first_type
 - __gnu_cxx::pair< _T1, _T2 >, 2721
 - __gnu_parallel::iterator_pair< _Iterator1, _Iterator2, _IteratorCategory >, 925
 - std::pair< _T1, _T2 >, 2727
- fixed
 - std, 606
 - std::basic_fstream< _CharT, _Traits >, 1189
 - std::basic_ifstream< _CharT, _Traits >, 1232
 - std::basic_ios< _CharT, _Traits >, 1256
 - std::basic_iostream< _CharT, _Traits >, 1303
 - std::basic_istream< _CharT, _Traits >, 1343
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1383
 - std::basic_ofstream< _CharT, _Traits >, 1417
 - std::basic_ostream< _CharT, _Traits >, 1448
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1481
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1766
 - std::ios_base, 2307
- flags
 - std::basic_fstream< _CharT, _Traits >, 1154
 - std::basic_ifstream< _CharT, _Traits >, 1205
- std::basic_ios< _CharT, _Traits >, 1246
- std::basic_iostream< _CharT, _Traits >, 1270
- std::basic_istream< _CharT, _Traits >, 1317
- std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1357, 1358
- std::basic_ofstream< _CharT, _Traits >, 1397
- std::basic_ostream< _CharT, _Traits >, 1430
- std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1463
- std::basic_regex< _Ch_type, _Rx_traits >, 1492
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1732, 1733
- std::ios_base, 2300
- flat_map, 3617
- flat_set, 3617
- flip
 - std::bitset< _Nb >, 1815
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2110
- float_denorm_style
 - std, 592
- float_round_style
 - std, 592
- floatfield
 - std::basic_fstream< _CharT, _Traits >, 1189
 - std::basic_ifstream< _CharT, _Traits >, 1232
 - std::basic_ios< _CharT, _Traits >, 1256
 - std::basic_iostream< _CharT, _Traits >, 1303
 - std::basic_istream< _CharT, _Traits >, 1343
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1383
 - std::basic_ofstream< _CharT, _Traits >, 1417
 - std::basic_ostream< _CharT, _Traits >, 1448
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1481
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1766
 - std::ios_base, 2307
- floor
 - Time, 378, 379
- flush
 - std, 606
 - std::basic_fstream< _CharT, _Traits >, 1155
 - std::basic_iostream< _CharT, _Traits >, 1271
 - std::basic_ofstream< _CharT, _Traits >, 1398
 - std::basic_ostream< _CharT, _Traits >, 1430
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1463
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1733
- fmtflags
 - std::basic_fstream< _CharT, _Traits >, 1148
 - std::basic_ifstream< _CharT, _Traits >, 1199
 - std::basic_ios< _CharT, _Traits >, 1240
 - std::basic_iostream< _CharT, _Traits >, 1265

- std::basic_istream< _CharT, _Traits >, 1312
- std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1352
- std::basic_ofstream< _CharT, _Traits >, 1391
- std::basic_ostream< _CharT, _Traits >, 1425
- std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1457
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1727
- std::ios_base, 2298
- for_each
 - Non-Mutating, 54
- for_each.h, 3668
- for_each_minimal_n
 - __gnu_parallel::Settings, 1040
- for_each_n
 - Non-Mutating, 54
- for_each_selectors.h, 3668
- format, 3617
 - std::filesystem::path, 2755
 - std::match_results< _Bi_iter, _Alloc >, 2450
- format_default
 - std::regex_constants, 728
- format_first_only
 - std::regex_constants, 729
- format_no_copy
 - std::regex_constants, 729
- format_sed
 - std::regex_constants, 729
- formatfwd.h, 3330
- formatter.h, 3515
- forward
 - Utilities, 307
- forward_as_tuple
 - Utilities, 308
- forward_like
 - Utilities, 309
- forward_list, 3617–3619
 - std::forward_list< _Tp, _Alloc >, 2180–2183
- forward_list.h, 3330
- forward_list.tcc, 3331
- fpos
 - std::fpos< _StateT >, 2197
- frac_digits
 - std::moneypunct< _CharT, _Intl >, 2494
 - std::moneypunct_byname< _CharT, _Intl >, 2502
- from_bytes
 - std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3281
- from_chars
 - std, 607
- front
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 865
- __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1560
- std::basic_string< _CharT, _Traits, _Alloc >, 1651
- std::chrono::tzdb_list, 3094
- std::deque< _Tp, _Alloc >, 2078, 2079
- std::forward_list< _Tp, _Alloc >, 2188
- std::list< _Tp, _Alloc >, 2376
- std::queue< _Tp, _Sequence >, 2795
- std::vector< _Tp, _Alloc >, 3245, 3246
- front_insert_iterator
 - std::front_insert_iterator< _Container >, 2201
- front_inserter
 - Iterators, 176
- fs_dir.h, 3331, 3332
- fs_fwd.h, 3332, 3334
- fs_ops.h, 3336, 3338
- fs_path.h, 3341
- fstream, 3619
 - I/O, 170
- fstream.tcc, 3341
- funcref_impl.h, 3342
- functexcept.h, 3342
- function
 - std::function< _Res(_ArgTypes...) >, 2203, 2204
- Function Objects, 316
 - mem_fn, 317
- function_ref
 - std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex) >, 2209, 2210
- functional, 3620, 3621
- functional_hash.h, 3342
- functions.h, 3515
- funcwrap.h, 3343
- future, 3623
 - std::future< _Res >, 2212
 - std::future< _Res & >, 2213
 - std::future< void >, 2215
- future_category
 - Futures, 121
- future_errc
 - Futures, 120
- future_status
 - Futures, 120
- Futures, 118
 - async, 120
 - future_category, 121
 - future_errc, 120
 - future_status, 120
 - launch, 120
 - make_error_code, 121
 - make_error_condition, 121
 - swap, 121
- gamma_distribution

- std::gamma_distribution< _RealType >, 2218
- gbump
 - __gnu_cxx::enc_filebuf< _CharT >, 2124
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2950
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2970
 - std::basic_filebuf< _CharT, _Traits >, 1125
 - std::basic_streambuf< _CharT, _Traits >, 1499
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1706
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3262
- gcd
 - std::experimental, 714
- gcount
 - std::basic_fstream< _CharT, _Traits >, 1155
 - std::basic_ifstream< _CharT, _Traits >, 1206
 - std::basic_iostream< _CharT, _Traits >, 1271
 - std::basic_istream< _CharT, _Traits >, 1317
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1358
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1733
- Generalized Numeric operations, 5
 - accumulate, 6, 7
 - adjacent_difference, 7, 8
 - exclusive_scan, 8, 9
 - inclusive_scan, 10, 11
 - inner_product, 12
 - iota, 13
 - partial_sum, 14
 - reduce, 15, 16
 - transform_exclusive_scan, 16
 - transform_inclusive_scan, 17, 18
 - transform_reduce, 19, 20
- generate
 - Mutating, 26
- generate_canonical
 - Random Number Generation, 238
 - std, 607
- generate_minimal_n
 - __gnu_parallel::Settings, 1040
- generate_n
 - Mutating, 27
- generator, 3624
- generic_category
 - Diagnostics, 130
- get
 - __gnu_parallel::Settings, 1039
 - std, 607–609
 - std::auto_ptr< _Tp >, 1103
 - std::basic_fstream< _CharT, _Traits >, 1155–1157
 - std::basic_ifstream< _CharT, _Traits >, 1206–1208
 - std::basic_iostream< _CharT, _Traits >, 1271–1273
 - std::basic_istream< _CharT, _Traits >, 1318–1320
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1358–1360
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1733–1735
 - std::future< _Res >, 2212
 - std::future< _Res & >, 2213
 - std::future< void >, 2215
 - std::money_get< _CharT, _InIter >, 2481, 2482
 - std::num_get< _CharT, _InIter >, 2585–2592
 - std::shared_future< _Res >, 2909
 - std::shared_future< _Res & >, 2910
 - std::shared_ptr< _Tp >, 2922
 - std::time_get< _CharT, _InIter >, 3024, 3025
 - std::time_get_byname< _CharT, _InIter >, 3035, 3036
 - std::unique_ptr< _Tp, _Dp >, 3110
 - std::unique_ptr< _Tp[], _Dp >, 3115
 - Utilities, 309
- get_actual_size
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2274
- get_allocator
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 865
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1561
 - std::basic_string< _CharT, _Traits, _Alloc >, 1652
 - std::deque< _Tp, _Alloc >, 2079
 - std::forward_list< _Tp, _Alloc >, 2188
 - std::list< _Tp, _Alloc >, 2376
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2434
 - std::match_results< _Bi_iter, _Alloc >, 2450
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2527
 - std::multiset< _Key, _Compare, _Alloc >, 2550
 - std::set< _Key, _Compare, _Alloc >, 2901
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2111
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3139
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3166
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3191
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3215
 - std::vector< _Tp, _Alloc >, 3246
 - std::vector< bool, _Alloc >, 3257
- get_child
 - __gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 965
 - __gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 965

- `_Alloc` >, 968
- `get_comb_hash_fn`
 - `__gnu_pbds::detail::cc_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1839
- `get_comb_probe_fn`
 - `__gnu_pbds::detail::gp_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2232
- `get_date`
 - `std::time_get` < _CharT, _InIter >, 3025
 - `std::time_get_byname` < _CharT, _InIter >, 3037
- `get_default_resource`
 - `memory_resource`, 3644
- `get_deleter`
 - Pointer Abstractions, 349
 - `std::experimental`, 714
 - `std::unique_ptr` < _Tp, _Dp >, 3110
 - `std::unique_ptr` < _Tp[], _Dp >, 3115
- `get_eq_fn`
 - `__gnu_pbds::detail::cc_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1840
 - `__gnu_pbds::detail::gp_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2233
- `get_hash_fn`
 - `__gnu_pbds::detail::cc_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1840
 - `__gnu_pbds::detail::gp_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2233
- `get_id`
 - `std::this_thread`, 733
- `get_l_child`
 - `__gnu_pbds::detail::bin_search_tree_const_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1775
 - `__gnu_pbds::detail::bin_search_tree_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1780
 - `__gnu_pbds::detail::ov_tree_node_const_it` < Value_Type, Metadata_Type, _Alloc >, 2710
 - `__gnu_pbds::detail::ov_tree_node_it` < Value_Type, Metadata_Type, _Alloc >, 2712
- `get_load`
 - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger` < External_Load_Access, Size_Type >, 1828
- `get_loads`
 - `__gnu_pbds::hash_load_check_resize_trigger` < External_Load_Access, Size_Type >, 2263
 - `__gnu_pbds::hash_standard_resize_policy` < Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2274
- `get_metadata`
 - `__gnu_pbds::detail::bin_search_tree_const_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1776
 - `__gnu_pbds::detail::bin_search_tree_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1780
 - `__gnu_pbds::detail::pat_trie_base::_Node_citer` < Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 965
 - `__gnu_pbds::detail::pat_trie_base::_Node_iter` < Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 968
- `get_money`
 - `std`, 609
- `get_monthname`
 - `std::time_get` < _CharT, _InIter >, 3026
 - `std::time_get_byname` < _CharT, _InIter >, 3037
- `get_nearest_larger_size`
 - `__gnu_pbds::sample_size_policy`, 2873
- `get_nearest_smaller_size`
 - `__gnu_pbds::sample_size_policy`, 2873
- `get_new_handler`
 - `std`, 609
- `get_new_size`
 - `__gnu_pbds::hash_standard_resize_policy` < Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2274
 - `__gnu_pbds::sample_resize_policy`, 2868
- `get_probe_fn`
 - `__gnu_pbds::detail::gp_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2233
- `get_r_child`
 - `__gnu_pbds::detail::bin_search_tree_const_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1776
 - `__gnu_pbds::detail::bin_search_tree_node_it` < Node, Const_Iterator, Iterator, _Alloc >, 1780
 - `__gnu_pbds::detail::ov_tree_node_const_it` < Value_Type, Metadata_Type, _Alloc >, 2710
 - `__gnu_pbds::detail::ov_tree_node_it` < Value_Type, Metadata_Type, _Alloc >, 2712
- `get_resize_policy`
 - `__gnu_pbds::detail::cc_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1840
 - `__gnu_pbds::detail::gp_ht_map` < Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2233, 2234
- `get_size_policy`
 - `__gnu_pbds::hash_standard_resize_policy` < Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >

- [>](#), [2275](#)
- get_temporary_buffer
 - std, [609](#)
- get_terminate
 - Exceptions, [135](#)
- get_time
 - std, [610](#)
 - std::time_get< _CharT, _InIter >, [3027](#)
 - std::time_get_byname< _CharT, _InIter >, [3038](#)
- get_trigger_policy
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, [2275](#)
- get_weekday
 - std::time_get< _CharT, _InIter >, [3027](#)
 - std::time_get_byname< _CharT, _InIter >, [3038](#)
- get_year
 - std::time_get< _CharT, _InIter >, [3028](#)
 - std::time_get_byname< _CharT, _InIter >, [3039](#)
- getline
 - std, [610–612](#)
 - std::basic_fstream< _CharT, _Traits >, [1158](#)
 - std::basic_ifstream< _CharT, _Traits >, [1209](#)
 - std::basic_iostream< _CharT, _Traits >, [1274](#)
 - std::basic_istream< _CharT, _Traits >, [1320](#), [1321](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1361](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1736](#)
- getloc
 - __gnu_cxx::enc_filebuf< _CharT >, [2124](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2950](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2970](#)
 - std::basic_filebuf< _CharT, _Traits >, [1125](#)
 - std::basic_fstream< _CharT, _Traits >, [1159](#)
 - std::basic_ifstream< _CharT, _Traits >, [1210](#)
 - std::basic_ios< _CharT, _Traits >, [1246](#)
 - std::basic_iostream< _CharT, _Traits >, [1275](#)
 - std::basic_istream< _CharT, _Traits >, [1321](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1362](#)
 - std::basic_ofstream< _CharT, _Traits >, [1398](#)
 - std::basic_ostream< _CharT, _Traits >, [1431](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1464](#)
 - std::basic_regex< _Ch_type, _Rx_traits >, [1492](#)
 - std::basic_streambuf< _CharT, _Traits >, [1500](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1706](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1737](#)
 - std::ios_base, [2300](#)
 - std::regex_traits< _Ch_type >, [2843](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3263](#)
- global
 - std::locale, [2393](#)
- good
 - std::basic_fstream< _CharT, _Traits >, [1159](#)
 - std::basic_ifstream< _CharT, _Traits >, [1210](#)
 - std::basic_ios< _CharT, _Traits >, [1246](#)
 - std::basic_iostream< _CharT, _Traits >, [1275](#)
 - std::basic_istream< _CharT, _Traits >, [1322](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1362](#)
 - std::basic_ofstream< _CharT, _Traits >, [1398](#)
 - std::basic_ostream< _CharT, _Traits >, [1431](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1464](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1737](#)
- goodbit
 - std::basic_fstream< _CharT, _Traits >, [1189](#)
 - std::basic_ifstream< _CharT, _Traits >, [1232](#)
 - std::basic_ios< _CharT, _Traits >, [1256](#)
 - std::basic_iostream< _CharT, _Traits >, [1303](#)
 - std::basic_istream< _CharT, _Traits >, [1343](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1383](#)
 - std::basic_ofstream< _CharT, _Traits >, [1417](#)
 - std::basic_ostream< _CharT, _Traits >, [1448](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1482](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1766](#)
 - std::ios_base, [2307](#)
- gp_hash_table
 - __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >, [2224–2228](#)
- gp_ht_map.hpp, [3573](#)
- gptr
 - __gnu_cxx::enc_filebuf< _CharT >, [2124](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2951](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2970](#)
 - std::basic_filebuf< _CharT, _Traits >, [1125](#)
 - std::basic_streambuf< _CharT, _Traits >, [1500](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1707](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3263](#)
- grep
 - std::regex_constants, [729](#)
- grouping
 - std::moneypunct< _CharT, _Intl >, [2495](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2502](#)
 - std::numpunct< _CharT >, [2687](#)
 - std::numpunct_byname< _CharT >, [2692](#)
- gslice

- std::numeric_limits< signed char >, 2657
- std::numeric_limits< unsigned char >, 2661
- std::numeric_limits< unsigned int >, 2665
- std::numeric_limits< unsigned long >, 2669
- std::numeric_limits< unsigned long long >, 2673
- std::numeric_limits< unsigned short >, 2677
- std::numeric_limits< wchar_t >, 2681
- hash
 - std::__cxx11::collate< _CharT >, 1906
- Hash-Based, 146
- hash_bytes.h, 3344
- hash_eq_fn.hpp, 3573
- hash_exponential_size_policy
 - __gnu_pbds::hash_exponential_size_policy< Size_Type >, 2261
- hash_exponential_size_policy_imp.hpp, 3585
- hash_fun.h, 3291
- hash_function
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3139
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3166
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3191
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3215
- hash_load_check_resize_trigger
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2263
- hash_load_check_resize_trigger_imp.hpp, 3585
- hash_load_check_resize_trigger_size_base.hpp, 3585
- hash_map, 3292
- hash_policy.hpp, 3592
- hash_prime_size_policy
 - __gnu_pbds::hash_prime_size_policy, 2270
- hash_prime_size_policy_imp.hpp, 3585
- hash_set, 3292
- hash_standard_resize_policy
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy_Type, Trigger_Policy, External_Size_Access, Size_Type >, 2274
- hash_standard_resize_policy_imp.hpp, 3585
- hasher
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3125
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3154
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
- Hashes, 324
- hashtable.h, 3344, 3345
- hashtable_policy.h, 3345
- Heap, 91
 - is_heap, 92
 - is_heap_until, 92, 93
 - make_heap, 93, 94
 - pop_heap, 94, 95
 - push_heap, 95
 - sort_heap, 96
- Heap-Based, 148
- helper_functions.h, 3517
- hermite
 - Mathematical Special Functions, 213
 - TR1 Mathematical Special Functions, 255
- hermitef
 - Mathematical Special Functions, 213
- hermitel
 - Mathematical Special Functions, 214
- hex
 - std, 612
 - std::basic_fstream< _CharT, _Traits >, 1189
 - std::basic_ifstream< _CharT, _Traits >, 1232
 - std::basic_ios< _CharT, _Traits >, 1257
 - std::basic_iostream< _CharT, _Traits >, 1304
 - std::basic_istream< _CharT, _Traits >, 1343
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1384
 - std::basic_ofstream< _CharT, _Traits >, 1417
 - std::basic_ostream< _CharT, _Traits >, 1449
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1482
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1766
 - std::ios_base, 2307
- hexfloat
 - std, 612
- high_resolution_clock
 - Time, 375
- hours
 - Time, 375
- hyperg
 - Mathematical Special Functions, 214
 - std::tr1, 738
- hypergf
 - Mathematical Special Functions, 214
 - std::tr1, 738
- hypergl
 - Mathematical Special Functions, 215
 - std::tr1, 739
- I/O, 168
 - filebuf, 170
 - fstream, 170
 - ifstream, 170
 - ios, 170
 - iostream, 170

- istream, 170
- istreamstream, 170
- ofstream, 170
- ostream, 170
- ostreamstream, 171
- streambuf, 171
- stringbuf, 171
- stringstream, 171
- wfilebuf, 171
- wfstream, 171
- wifstream, 171
- wios, 171
- wiostream, 171
- wistream, 171
- wistreamstream, 171
- wofstream, 172
- wostream, 172
- wostringstream, 172
- wstreambuf, 172
- wstringbuf, 172
- wstringstream, 172
- icase
 - std::regex_constants, 729
- id
 - std::__cxx11::collate< _CharT >, 1907
 - std::ctype< _CharT >, 1965
 - std::ctype< char >, 1995
 - std::ctype< wchar_t >, 2009
 - std::ctype_byname< _CharT >, 2024
 - std::ctype_byname< char >, 2047
 - std::locale::id, 2277
 - std::messages< _CharT >, 2469
 - std::messages_byname< _CharT >, 2472
 - std::money_get< _CharT, _InIter >, 2483
 - std::money_put< _CharT, _OutIter >, 2488
 - std::moneypunct< _CharT, _Intl >, 2497
 - std::moneypunct_byname< _CharT, _Intl >, 2504
 - std::num_get< _CharT, _InIter >, 2593
 - std::num_put< _CharT, _OutIter >, 2606
 - std::numpunct< _CharT >, 2688
 - std::numpunct_byname< _CharT >, 2693
 - std::time_get< _CharT, _InIter >, 3029
 - std::time_get_byname< _CharT, _InIter >, 3040
 - std::time_put< _CharT, _OutIter >, 3045
 - std::time_put_byname< _CharT, _OutIter >, 3048
- identity_element
 - SGL, 160
- ifstream
 - I/O, 170
- ignore
 - std, 657
 - std::basic_fstream< _CharT, _Traits >, 1160
 - std::basic_ifstream< _CharT, _Traits >, 1210, 1211
 - std::basic_iostream< _CharT, _Traits >, 1275, 1276
 - std::basic_istream< _CharT, _Traits >, 1322, 1323
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1362, 1363
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1738
- imbue
 - __gnu_cxx::enc_filebuf< _CharT >, 2125
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2951
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2971
 - std::basic_filebuf< _CharT, _Traits >, 1126
 - std::basic_fstream< _CharT, _Traits >, 1161
 - std::basic_ifstream< _CharT, _Traits >, 1211
 - std::basic_ios< _CharT, _Traits >, 1247
 - std::basic_iostream< _CharT, _Traits >, 1276
 - std::basic_istream< _CharT, _Traits >, 1323
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1364
 - std::basic_ofstream< _CharT, _Traits >, 1398
 - std::basic_ostream< _CharT, _Traits >, 1431
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1464
 - std::basic_regex< _Ch_type, _Rx_traits >, 1492
 - std::basic_streambuf< _CharT, _Traits >, 1500
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1707
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1739
 - std::ios_base, 2301
 - std::regex_traits< _Ch_type >, 2843
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3263
- in
 - std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 769
 - std::basic_fstream< _CharT, _Traits >, 1189
 - std::basic_ifstream< _CharT, _Traits >, 1232
 - std::basic_ios< _CharT, _Traits >, 1257
 - std::basic_iostream< _CharT, _Traits >, 1304
 - std::basic_istream< _CharT, _Traits >, 1343
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1384
 - std::basic_ofstream< _CharT, _Traits >, 1417
 - std::basic_ostream< _CharT, _Traits >, 1449
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1482
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1766
 - std::codecvt< _InternT, _ExternT, _StateT >, 1856
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1861, 1862
 - std::codecvt< char, char, mbstate_t >, 1869
 - std::codecvt< char16_t, char, mbstate_t >, 1876, 1877
 - std::codecvt< char32_t, char, mbstate_t >, 1883, 1884

- std::codecvt< wchar_t, char, mbstate_t >, [1891](#), [1892](#)
- std::codecvt_byname< _InternT, _ExternT, _StateT >, [1900](#)
- std::ios_base, [2307](#)
- in_avail
 - __gnu_cxx::enc_filebuf< _CharT >, [2125](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2951](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2971](#)
 - std::basic_filebuf< _CharT, _Traits >, [1126](#)
 - std::basic_streambuf< _CharT, _Traits >, [1501](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1707](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3264](#)
- in_place
 - Optional values, [294](#)
- includes
 - Set Operations, [98](#)
- inclusive_scan
 - Generalized Numeric operations, [10](#), [11](#)
- increment
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, [2362](#)
- independent_bits_engine
 - std::independent_bits_engine< __RandomNumberEngine, __w, _UIntType >, [2280](#), [2281](#)
- index_sequence
 - std, [590](#)
- index_sequence_for
 - std, [590](#)
- indirect.h, [3345](#)
- indirect_array
 - Numeric Arrays, [224](#)
- indirect_array.h, [3345](#)
- infinity
 - std::numeric_limits< _Tp >, [2607](#)
- info_fn_imps.hpp, [3560](#), [3561](#)
- init
 - std::basic_fstream< _CharT, _Traits >, [1161](#)
 - std::basic_ifstream< _CharT, _Traits >, [1212](#)
 - std::basic_ios< _CharT, _Traits >, [1247](#)
 - std::basic_iostream< _CharT, _Traits >, [1277](#)
 - std::basic_istream< _CharT, _Traits >, [1324](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1364](#)
 - std::basic_ofstream< _CharT, _Traits >, [1399](#)
 - std::basic_ostream< _CharT, _Traits >, [1431](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1464](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1739](#)
- initializer_list, [3739](#)
- inner_product
 - Generalized Numeric operations, [12](#)
- inout_ptr
 - std, [612](#)
- inplace_merge
 - Sorting, [66](#)
- inplace_vector, [3624](#)
- insert
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [865–870](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1561–1567](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1652–1661](#)
 - std::deque< _Tp, _Alloc >, [2079–2081](#)
 - std::list< _Tp, _Alloc >, [2376–2378](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2434–2437](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2527–2530](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2550](#), [2551](#)
 - std::set< _Key, _Compare, _Alloc >, [2902](#), [2903](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3139–3142](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3166–3169](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3191–3193](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3215–3218](#)
 - std::vector< _Tp, _Alloc >, [3246–3248](#)
- insert_after
 - std::forward_list< _Tp, _Alloc >, [2188](#), [2189](#)
- insert_fn_imps.hpp, [3561](#), [3562](#)
- insert_iterator
 - std::insert_iterator< _Container >, [2291](#)
- insert_join_fn_imps.hpp, [3581](#)
- insert_no_store_hash_fn_imps.hpp, [3570](#), [3571](#)
- insert_store_hash_fn_imps.hpp, [3571](#)
- inserter
 - Iterators, [177](#)
- int_type
 - std::basic_ios< _CharT, _Traits >, [1241](#)
 - std::basic_streambuf< _CharT, _Traits >, [1497](#)
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3260](#)
- intcmp.h, [3346](#)
- internal
 - std, [613](#)
 - std::basic_fstream< _CharT, _Traits >, [1189](#)
 - std::basic_ifstream< _CharT, _Traits >, [1233](#)
 - std::basic_ios< _CharT, _Traits >, [1257](#)
 - std::basic_iostream< _CharT, _Traits >, [1304](#)
 - std::basic_istream< _CharT, _Traits >, [1343](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1384](#)

- std::basic_ofstream< _CharT, _Traits >, 1417
- std::basic_ostream< _CharT, _Traits >, 1449
- std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1482
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1767
- std::ios_base, 2308
- intervals
 - std::piecewise_constant_distribution< _RealType >, 2762
 - std::piecewise_linear_distribution< _RealType >, 2766
- intl
 - std::moneypunct< _CharT, _Intl >, 2497
- Invalidation Guarantees, 151
- invoke.h, 3346
- io_errc
 - std, 592
- iomanip, 3625
- ios, 3626
 - I/O, 170
- ios_base.h, 3347
- iosfwd, 3627
- iosfwd_file.h, 3349
- iosfwd_span.h, 3349
- iosfwd_string.h, 3349
- iosfwd_sync.h, 3349
- iostate
 - std::basic_fstream< _CharT, _Traits >, 1149
 - std::basic_ifstream< _CharT, _Traits >, 1200
 - std::basic_ios< _CharT, _Traits >, 1241
 - std::basic_iostream< _CharT, _Traits >, 1266
 - std::basic_istream< _CharT, _Traits >, 1312
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1352
 - std::basic_ofstream< _CharT, _Traits >, 1392
 - std::basic_ostream< _CharT, _Traits >, 1425
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1458
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1728
 - std::ios_base, 2299
- iostream, 3627
 - I/O, 170
- iota
 - __gnu_cxx, 404
 - Generalized Numeric operations, 13
- is
 - std::__ctype_abstract_base< _CharT >, 782, 783
 - std::ctype< _CharT >, 1959, 1960
 - std::ctype< char >, 1980–1982
 - std::ctype< wchar_t >, 2004
 - std::ctype_byname< _CharT >, 2018, 2019
 - std::ctype_byname< char >, 2037, 2038
- is_always_equal
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 758
 - std::allocator_traits< _Alloc >, 1064
 - std::allocator_traits< allocator< _Tp > >, 1069
 - std::allocator_traits< allocator< void > >, 1074
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
- is_am
 - Time, 379
- is_bind_expression_v
 - std::experimental, 715
- is_bounded
 - std::__numeric_limits_base, 816
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621
 - std::numeric_limits< char32_t >, 2625
 - std::numeric_limits< double >, 2629
 - std::numeric_limits< float >, 2633
 - std::numeric_limits< int >, 2637
 - std::numeric_limits< long >, 2641
 - std::numeric_limits< long double >, 2645
 - std::numeric_limits< long long >, 2649
 - std::numeric_limits< short >, 2653
 - std::numeric_limits< signed char >, 2657
 - std::numeric_limits< unsigned char >, 2661
 - std::numeric_limits< unsigned int >, 2665
 - std::numeric_limits< unsigned long >, 2669
 - std::numeric_limits< unsigned long long >, 2673
 - std::numeric_limits< unsigned short >, 2677
 - std::numeric_limits< wchar_t >, 2681
- is_corresponding_member
 - Metaprogramming, 362
- is_detected
 - Detection idiom, 291
- is_detected_convertible
 - Detection idiom, 292
- is_detected_convertible_v
 - Detection idiom, 292
- is_detected_exact
 - Detection idiom, 292
- is_detected_exact_v
 - Detection idiom, 292
- is_detected_v
 - Detection idiom, 292
- is_emptyset
 - std::tr2::bool_set, 1822
- is_exact
 - std::__numeric_limits_base, 816
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621

- std::numeric_limits< char32_t >, 2625
- std::numeric_limits< double >, 2629
- std::numeric_limits< float >, 2633
- std::numeric_limits< int >, 2637
- std::numeric_limits< long >, 2641
- std::numeric_limits< long double >, 2645
- std::numeric_limits< long long >, 2649
- std::numeric_limits< short >, 2653
- std::numeric_limits< signed char >, 2657
- std::numeric_limits< unsigned char >, 2661
- std::numeric_limits< unsigned int >, 2665
- std::numeric_limits< unsigned long >, 2669
- std::numeric_limits< unsigned long long >, 2673
- std::numeric_limits< unsigned short >, 2677
- std::numeric_limits< wchar_t >, 2681
- is_grow_needed
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger
 - External_Load_Access, Size_Type >, 1828
 - __gnu_pbds::sample_resize_trigger, 2871
- is_heap
 - Heap, 92
- is_heap_until
 - Heap, 92, 93
- is_iec559
 - std::__numeric_limits_base, 816
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621
 - std::numeric_limits< char32_t >, 2625
 - std::numeric_limits< double >, 2629
 - std::numeric_limits< float >, 2633
 - std::numeric_limits< int >, 2637
 - std::numeric_limits< long >, 2641
 - std::numeric_limits< long double >, 2645
 - std::numeric_limits< long long >, 2649
 - std::numeric_limits< short >, 2653
 - std::numeric_limits< signed char >, 2657
 - std::numeric_limits< unsigned char >, 2661
 - std::numeric_limits< unsigned int >, 2665
 - std::numeric_limits< unsigned long >, 2669
 - std::numeric_limits< unsigned long long >, 2673
 - std::numeric_limits< unsigned short >, 2677
 - std::numeric_limits< wchar_t >, 2681
- is_indeterminate
 - std::tr2::bool_set, 1822
- is_integer
 - std::__numeric_limits_base, 816
 - std::numeric_limits< _Tp >, 2609
 - std::numeric_limits< bool >, 2613
 - std::numeric_limits< char >, 2617
 - std::numeric_limits< char16_t >, 2621
 - std::numeric_limits< char32_t >, 2625
 - std::numeric_limits< double >, 2629
- std::numeric_limits< float >, 2633
- std::numeric_limits< int >, 2637
- std::numeric_limits< long >, 2641
- std::numeric_limits< long double >, 2645
- std::numeric_limits< long long >, 2649
- std::numeric_limits< short >, 2653
- std::numeric_limits< signed char >, 2657
- std::numeric_limits< unsigned char >, 2661
- std::numeric_limits< unsigned int >, 2665
- std::numeric_limits< unsigned long >, 2669
- std::numeric_limits< unsigned long long >, 2673
- std::numeric_limits< unsigned short >, 2677
- std::numeric_limits< wchar_t >, 2681
- is_layout_compatible_v
 - Metaprogramming, 363
 - Variable Templates, 364
- is_modulo
 - std::__numeric_limits_base, 816
 - std::numeric_limits< _Tp >, 2610
 - std::numeric_limits< bool >, 2614
 - std::numeric_limits< char >, 2618
 - std::numeric_limits< char16_t >, 2622
 - std::numeric_limits< char32_t >, 2626
 - std::numeric_limits< double >, 2630
 - std::numeric_limits< float >, 2634
 - std::numeric_limits< int >, 2638
 - std::numeric_limits< long >, 2642
 - std::numeric_limits< long double >, 2646
 - std::numeric_limits< long long >, 2650
 - std::numeric_limits< short >, 2654
 - std::numeric_limits< signed char >, 2658
 - std::numeric_limits< unsigned char >, 2662
 - std::numeric_limits< unsigned int >, 2666
 - std::numeric_limits< unsigned long >, 2670
 - std::numeric_limits< unsigned long long >, 2674
 - std::numeric_limits< unsigned short >, 2678
 - std::numeric_limits< wchar_t >, 2682
- is_open
 - __gnu_cxx::enc_filebuf< _CharT >, 2125
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2952
 - std::basic_filebuf< _CharT, _Traits >, 1126
 - std::basic_fstream< _CharT, _Traits >, 1162
 - std::basic_ifstream< _CharT, _Traits >, 1212
 - std::basic_ofstream< _CharT, _Traits >, 1399
- is_partitioned
 - Mutating, 27
- is_permutation
 - Non-Mutating, 55, 56
- is_placeholder_v
 - std::experimental, 715
- is_pm
 - Time, 379
- is_pointer_interconvertible_base_of_v
 - Metaprogramming, 363

- Variable Templates, [364](#)
- is_pointer_interconvertible_with_class
 - Metaprogramming, [362](#)
- is_resize_needed
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger
 - External_Load_Access, Size_Type >, [1828](#)
 - __gnu_pbds::sample_resize_policy, [2868](#)
 - __gnu_pbds::sample_resize_trigger, [2871](#)
- is_signed
 - std::__numeric_limits_base, [816](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- is_singleton
 - std::tr2::bool_set, [1822](#)
- is_sorted
 - Sorting, [67](#)
- is_sorted_until
 - Sorting, [68](#)
- is_specialized
 - std::__numeric_limits_base, [816](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- std::numeric_limits< unsigned short >, [2678](#)
- std::numeric_limits< wchar_t >, [2682](#)
- is_sufficiently_aligned
 - Memory, [328](#)
- isalnum
 - std, [613](#)
- isalpha
 - std, [613](#)
- isblank
 - std, [613](#)
- isctrl
 - std, [614](#)
- isctype
 - std::regex_traits< _Ch_type >, [2844](#)
- isdigit
 - std, [614](#)
- isgraph
 - std, [614](#)
- islower
 - std, [614](#)
- isprint
 - std, [614](#)
- ispunct
 - std, [614](#)
- isspace
 - std, [615](#)
- istream, [3627](#)
 - I/O, [170](#)
- istream.tcc, [3350](#)
- istream_iterator
 - std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, [2331](#), [2332](#)
- istream_type
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
- istreambuf_iterator
 - std::istreambuf_iterator< _CharT, _Traits >, [2335](#)
- istringstream
 - I/O, [170](#)
- isupper
 - std, [615](#)
- isxdigit
 - std, [615](#)
- iter_rvalue_reference_t
 - std, [591](#)
- iter_swap
 - Mutating, [28](#)
- iter_type
 - std::money_get< _CharT, _InIter >, [2480](#)
 - std::money_put< _CharT, _OutIter >, [2484](#)
 - std::num_get< _CharT, _InIter >, [2578](#)
 - std::num_put< _CharT, _OutIter >, [2595](#)
 - std::time_get< _CharT, _InIter >, [3019](#)
 - std::time_put< _CharT, _OutIter >, [3042](#)
- iterator, [3628](#), [3629](#)

- std::set< _Key, _Compare, _Alloc >, [2891](#)
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3125](#)
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3154](#)
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3179](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3203](#)
- Iterator Tags, [178](#)
- iterator.h, [3669](#)
- iterator.hpp, [3591](#)
- iterator_category
 - __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, [1775](#)
 - __gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >, [1779](#)
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, [1789](#)
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, [1793](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, [2345](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, [2349](#)
 - std::back_insert_iterator< _Container >, [1107](#)
 - std::front_insert_iterator< _Container >, [2201](#)
 - std::insert_iterator< _Container >, [2291](#)
 - std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, [2331](#)
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
 - std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, [2339](#)
 - std::ostream_iterator< _Tp, _CharT, _Traits >, [2698](#)
 - std::ostreambuf_iterator< _CharT, _Traits >, [2702](#)
 - std::raw_storage_iterator< _OutputIterator, _Tp >, [2814](#)
- iterator_concepts.h, [3350](#)
- iterator_fn_imps.hpp, [3574](#)
- Iterators, [172](#)
 - back_inserter, [176](#)
 - front_inserter, [176](#)
 - inserter, [177](#)
 - make_reverse_iterator, [177](#)
 - operator==, [177](#)
- iterators_fn_imps.hpp, [3562](#), [3563](#)
- iword
 - std::basic_fstream< _CharT, _Traits >, [1162](#)
 - std::basic_ifstream< _CharT, _Traits >, [1212](#)
 - std::basic_ios< _CharT, _Traits >, [1247](#)
 - std::basic_iostream< _CharT, _Traits >, [1277](#)
 - std::basic_istream< _CharT, _Traits >, [1324](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1364](#)
- std::basic_ofstream< _CharT, _Traits >, [1399](#)
- std::basic_ostream< _CharT, _Traits >, [1432](#)
- std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1465](#)
- std::basic_stringstream< _CharT, _Traits, _Alloc >, [1740](#)
- std::ios_base, [2301](#)
- k
 - std::negative_binomial_distribution< _IntType >, [2562](#)
- key_comp
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2437](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2530](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2552](#)
 - std::set< _Key, _Compare, _Alloc >, [2904](#)
- key_compare
 - std::set< _Key, _Compare, _Alloc >, [2891](#)
- key_eq
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3143](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3170](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3194](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3219](#)
- key_equal
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3125](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3154](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3179](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3203](#)
- key_type
 - std::set< _Key, _Compare, _Alloc >, [2892](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3125](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3154](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- kill_dependency
 - Atomics, [116](#)
- L1_cache_size
 - __gnu_parallel::Settings, [1040](#)
- L2_cache_size
 - __gnu_parallel::Settings, [1041](#)
- laguerre

- Mathematical Special Functions, [215](#)
- TR1 Mathematical Special Functions, [255](#)
- laguerref
 - Mathematical Special Functions, [216](#)
- laguerrel
 - Mathematical Special Functions, [216](#)
- lambda
 - std::exponential_distribution< _RealType >, [2158](#)
- latch, [3629](#)
- launch
 - Futures, [120](#)
- lcm
 - std::experimental, [714](#)
- left
 - std, [615](#)
 - std::basic_fstream< _CharT, _Traits >, [1190](#)
 - std::basic_ifstream< _CharT, _Traits >, [1233](#)
 - std::basic_ios< _CharT, _Traits >, [1257](#)
 - std::basic_iostream< _CharT, _Traits >, [1304](#)
 - std::basic_istream< _CharT, _Traits >, [1344](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1384](#)
 - std::basic_ofstream< _CharT, _Traits >, [1418](#)
 - std::basic_ostream< _CharT, _Traits >, [1449](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1482](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1767](#)
 - std::ios_base, [2308](#)
- left_child_next_sibling_heap.hpp, [3577](#)
- left_child_next_sibling_heap_const_iterator_
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_
 - Node, _Alloc >, [2345](#), [2346](#)
- left_child_next_sibling_heap_node_point_const_iterator_
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_
 - Node, _Alloc >, [2350](#)
- legendre
 - Mathematical Special Functions, [216](#)
 - TR1 Mathematical Special Functions, [255](#)
- legendref
 - Mathematical Special Functions, [217](#)
- legendrel
 - Mathematical Special Functions, [217](#)
- length
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [871](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1568](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1662](#)
 - std::match_results< _Bi_iter, _Alloc >, [2451](#)
 - std::regex_traits< _Ch_type >, [2844](#)
 - std::sub_match< _Biter >, [2993](#)
- lexicographical_compare
 - Sorting, [69](#)
- lexicographical_compare_3way
 - SGL, [160](#)
- lexicographical_compare_three_way
 - Sorting, [70](#)
- lfts_config.h, [3540](#)
- Library Fundamentals TS, [286](#)
- limits, [3629](#)
- linear_congruential_engine
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, [2359](#)
- linear_probe_fn_imp.hpp, [3574](#)
- list, [3631](#), [3632](#)
 - std::list< _Tp, _Alloc >, [2370](#), [2371](#)
- List-Based, [149](#)
- list.tcc, [3351](#)
- list_partition
 - __gnu_parallel, [463](#)
- list_partition.h, [3669](#)
- list_update
 - __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, [2388](#)
- list_update_policy.hpp, [3593](#)
- load_factor
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3143](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3170](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3194](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3219](#)
- local_iterator
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3125](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3154](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- local_time_format
 - Time, [380](#)
- locale, [3632](#)
 - std::locale, [2390](#)–[2392](#)
- locale_classes.h, [3352](#)
- locale_classes.tcc, [3352](#)
- locale_conv.h, [3353](#)
- locale_facets.h, [3353](#)
- locale_facets.tcc, [3354](#)
- locale_facets_nonio.h, [3355](#)
- locale_facets_nonio.tcc, [3356](#)
- localefwd.h, [3356](#)
- Locales, [178](#)
 - has_facet, [179](#)

- use_facet, [179](#)
- lock
 - Mutexes, [123](#)
- log
 - Complex Numbers, [186](#)
 - math.h, [3636](#)
- log10
 - Complex Numbers, [187](#)
 - math.h, [3636](#)
- logic_error
 - std::logic_error, [2399](#)
- Logical operator traits, [293](#)
- lookup_classname
 - std::regex_traits< _Ch_type >, [2845](#)
- lookup_collatename
 - std::regex_traits< _Ch_type >, [2846](#)
- losertree.h, [3669](#)
- lower_bound
 - Binary Search, [88, 89](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2437, 2438](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2530, 2531](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2552](#)
 - std::set< _Key, _Compare, _Alloc >, [2904](#)
- lowest
 - std::numeric_limits< _Tp >, [2608](#)
- lu_counter_metadata.hpp, [3580](#)
- lu_map.hpp, [3579](#)
- macros.h, [3518](#)
 - _GLIBCXX_DEBUG_VERIFY_AT_F, [3521](#)
 - _glibcxx_check_erase, [3519](#)
 - _glibcxx_check_erase_after, [3519](#)
 - _glibcxx_check_erase_range, [3520](#)
 - _glibcxx_check_erase_range_after, [3520](#)
 - _glibcxx_check_heap_pred, [3520](#)
 - _glibcxx_check_insert, [3520](#)
 - _glibcxx_check_insert_after, [3520](#)
 - _glibcxx_check_insert_range, [3520](#)
 - _glibcxx_check_insert_range_after, [3521](#)
 - _glibcxx_check_partitioned_lower, [3521](#)
 - _glibcxx_check_partitioned_lower_pred, [3521](#)
 - _glibcxx_check_partitioned_upper_pred, [3521](#)
 - _glibcxx_check_sorted_pred, [3521](#)
- make12
 - Time, [380](#)
- make24
 - Time, [380](#)
- make_array
 - Array creation functions, [288](#)
- make_boyer_moore_horspool_searcher
 - std::experimental, [714](#)
- make_boyer_moore_searcher
 - std::experimental, [714](#)
- make_default_searcher
 - std::experimental, [714](#)
- make_error_code
 - Diagnostics, [130](#)
 - Futures, [121](#)
- make_error_condition
 - Diagnostics, [130](#)
 - Futures, [121](#)
- make_exception_ptr
 - Exceptions, [136](#)
- make_heap
 - Heap, [93, 94](#)
- make_index_sequence
 - std, [591](#)
- make_integer_sequence
 - std, [591](#)
- make_ostream_joiner
 - std::experimental, [715](#)
- make_pair
 - Utilities, [310](#)
- make_reverse_iterator
 - Iterators, [177](#)
- make_shared
 - Pointer Abstractions, [349](#)
- make_signed_t
 - Metaprogramming, [361](#)
- make_tuple
 - Utilities, [310](#)
- make_unsigned_t
 - Metaprogramming, [361](#)
- malloc_allocator.h, [3547](#)
- map, [3632, 3633](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2423–2425](#)
- map.h, [3522](#)
- mapped_type
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3125](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3154](#)
- mark_count
 - std::basic_regex< _Ch_type, _Rx_traits >, [1492](#)
- mask_array
 - Numeric Arrays, [224](#)
- mask_array.h, [3357](#)
- mask_based_range_hashing.hpp, [3575](#)
- match_any
 - std::regex_constants, [729](#)
- match_continuous
 - std::regex_constants, [729](#)
- match_default
 - std::regex_constants, [730](#)
- match_flag_type

std::regex_constants, 723
 match_not_bol
 std::regex_constants, 730
 match_not_bow
 std::regex_constants, 730
 match_not_eol
 std::regex_constants, 730
 match_not_eow
 std::regex_constants, 730
 match_not_null
 std::regex_constants, 730
 match_prev_avail
 std::regex_constants, 730
 match_results
 std::match_results< _Bi_iter, _Alloc >, 2448
 math.h, 3633
 abs, 3635
 cos, 3635
 cosh, 3635
 exp, 3635
 fabs, 3636
 log, 3636
 log10, 3636
 pow, 3636
 sin, 3636
 sinh, 3636
 sqrt, 3636
 tan, 3636
 tanh, 3637
 Mathematical Special Functions, 192
 airy_ai, 196
 airy_aif, 196
 airy_ail, 196
 airy_bi, 196
 airy_bif, 196
 airy_bil, 197
 assoc_laguerre, 197
 assoc_laguerref, 198
 assoc_laguerrel, 198
 assoc_legendre, 198
 assoc_legendref, 199
 assoc_legendrel, 199
 beta, 199
 betaf, 200
 betal, 200
 comp_ellint_1, 200
 comp_ellint_1f, 201
 comp_ellint_1l, 201
 comp_ellint_2, 201
 comp_ellint_2f, 202
 comp_ellint_2l, 202
 comp_ellint_3, 202
 comp_ellint_3f, 203
 comp_ellint_3l, 203
 conf_hyperrg, 204
 conf_hyperrgf, 204
 conf_hyperrgl, 204
 cyl_bessel_i, 204
 cyl_bessel_if, 205
 cyl_bessel_il, 205
 cyl_bessel_j, 205
 cyl_bessel_jf, 206
 cyl_bessel_jl, 206
 cyl_bessel_k, 206
 cyl_bessel_kf, 207
 cyl_bessel_kl, 207
 cyl_neumann, 208
 cyl_neumannf, 208
 cyl_neumannl, 208
 ellint_1, 209
 ellint_1f, 209
 ellint_1l, 209
 ellint_2, 210
 ellint_2f, 210
 ellint_2l, 211
 ellint_3, 211
 ellint_3f, 212
 ellint_3l, 212
 expint, 212
 expintf, 213
 expintl, 213
 hermite, 213
 hermitef, 213
 hermitel, 214
 hyperrg, 214
 hyperrgf, 214
 hyperrgl, 215
 laguerre, 215
 laguerref, 216
 laguerrel, 216
 legendre, 216
 legendref, 217
 legendrel, 217
 riemann_zeta, 217
 riemann_zetaf, 218
 riemann_zetal, 218
 sph_bessel, 218
 sph_besself, 219
 sph_bessell, 219
 sph_legendre, 219
 sph_legendref, 220
 sph_legendrel, 220
 sph_neumann, 220
 sph_neumannf, 221
 sph_neumannl, 221
 max
 __gnu_parallel, 463
 Numeric Arrays, 228

- Sorting, [70](#), [71](#)
- std::bernoulli_distribution, [1770](#)
- std::binomial_distribution< _IntType >, [1800](#)
- std::cauchy_distribution< _RealType >, [1825](#)
- std::chi_squared_distribution< _RealType >, [1851](#)
- std::discard_block_engine< _RandomNumberEngine, __p, __r >, [2093](#)
- std::discrete_distribution< _IntType >, [2097](#)
- std::exponential_distribution< _RealType >, [2158](#)
- std::extreme_value_distribution< _RealType >, [2162](#)
- std::fisher_f_distribution< _RealType >, [2172](#)
- std::gamma_distribution< _RealType >, [2218](#)
- std::geometric_distribution< _IntType >, [2222](#)
- std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, [2282](#)
- std::linear_congruential_engine< _UIntType, __a, __c, __m >, [2359](#)
- std::lognormal_distribution< _RealType >, [2407](#)
- std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, [2464](#)
- std::negative_binomial_distribution< _IntType >, [2562](#)
- std::normal_distribution< _RealType >, [2569](#)
- std::numeric_limits< _Tp >, [2608](#)
- std::philox_engine< _UIntType, __w, __n, __r, __consts >, [2758](#)
- std::piecewise_constant_distribution< _RealType >, [2762](#)
- std::piecewise_linear_distribution< _RealType >, [2766](#)
- std::poisson_distribution< _IntType >, [2778](#)
- std::shuffle_order_engine< _RandomNumberEngine, __k >, [2926](#)
- std::student_t_distribution< _RealType >, [2988](#)
- std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, [2997](#)
- std::uniform_int_distribution< _IntType >, [3101](#)
- std::uniform_real_distribution< _RealType >, [3104](#)
- std::weibull_distribution< _RealType >, [3278](#)
- max_blocks_per_chunk
 - std::pmr::pool_options, [2781](#)
- max_bucket_count
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3143](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3170](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3194](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3219](#)
- max_count
 - __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, [2411](#)
- max_digits10
 - std::__numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- max_element
 - Sorting, [71](#), [72](#)
- max_element_minimal_n
 - __gnu_parallel::Settings, [1041](#)
- max_exponent
 - std::__numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- max_exponent10
 - std::__numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)

- std::numeric_limits< float >, 2634
- std::numeric_limits< int >, 2638
- std::numeric_limits< long >, 2642
- std::numeric_limits< long double >, 2646
- std::numeric_limits< long long >, 2650
- std::numeric_limits< short >, 2654
- std::numeric_limits< signed char >, 2658
- std::numeric_limits< unsigned char >, 2662
- std::numeric_limits< unsigned int >, 2666
- std::numeric_limits< unsigned long >, 2670
- std::numeric_limits< unsigned long long >, 2674
- std::numeric_limits< unsigned short >, 2678
- std::numeric_limits< wchar_t >, 2682
- max_load_factor
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3143
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3170
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3194
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3219
- max_size
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 762
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 871
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1568
 - std::allocator_traits< _Alloc >, 1067
 - std::allocator_traits< allocator< _Tp > >, 1072
 - std::allocator_traits< allocator< void > >, 1076
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1081
 - std::basic_string< _CharT, _Traits, _Alloc >, 1662
 - std::deque< _Tp, _Alloc >, 2081
 - std::forward_list< _Tp, _Alloc >, 2190
 - std::list< _Tp, _Alloc >, 2379
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2438
 - std::match_results< _Bi_iter, _Alloc >, 2451
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2531
 - std::multiset< _Key, _Compare, _Alloc >, 2553
 - std::set< _Key, _Compare, _Alloc >, 2905
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2111
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3144
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3194
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3220
 - std::vector< _Tp, _Alloc >, 3248
- max_size_type.h, 3357
- mdspan, 3637
- mean
 - std::normal_distribution< _RealType >, 2569
 - std::poisson_distribution< _IntType >, 2778
- mem_fn
 - Function Objects, 317
- Memory, 326
 - align, 327
 - assume_aligned, 328
 - is_sufficiently_aligned, 328
 - uninitialized_copy, 328
 - uninitialized_copy_n, 329
 - uninitialized_default_construct, 329
 - uninitialized_default_construct_n, 330
 - uninitialized_fill, 330
 - uninitialized_fill_n, 330
 - uninitialized_move, 331
 - uninitialized_move_n, 331
 - uninitialized_value_construct, 332
 - uninitialized_value_construct_n, 332
- memory, 3641–3643
- memory_order
 - Atomics, 116
- memory_resource, 3643, 3644
 - get_default_resource, 3644
 - null_memory_resource, 3644
 - set_default_resource, 3644
- memory_resource.h, 3357
- memoryfwd.h, 3358
- merge
 - Sorting, 72, 73
 - std::forward_list< _Tp, _Alloc >, 2190
 - std::list< _Tp, _Alloc >, 2379
- merge.h, 3670
- merge_minimal_n
 - __gnu_parallel::Settings, 1041
- merge_oversampling
 - __gnu_parallel::Settings, 1041
- mersenne_twister_engine
 - std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2464
- message
 - std::error_category, 2148
 - std::error_code, 2150
 - std::error_condition, 2152
- messages
 - std::locale, 2397
 - std::messages< _CharT >, 2468, 2469
- messages_members.h, 3734
- metadata_const_reference
 - __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1775

- __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1779
 - __gnu_pbds::detail::pat_trie_base::_Node_citer_< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 965
 - __gnu_pbds::detail::pat_trie_base::_Node_iter_< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 968
- metadata_reference
 - __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2410
 - __gnu_pbds::lu_move_to_front_policy< _Alloc >, 2415
- metadata_type
 - __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1775
 - __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1780
 - __gnu_pbds::detail::pat_trie_base::_Node_citer_< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 965
 - __gnu_pbds::detail::pat_trie_base::_Node_iter_< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 968
 - __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2410
 - __gnu_pbds::lu_move_to_front_policy< _Alloc >, 2415
 - __gnu_pbds::sample_update_policy, 2876
- Metaprogramming, 355
 - add_lvalue_reference_t, 360
 - add_pointer_t, 360
 - add_rvalue_reference_t, 360
 - aligned_storage_t, 360
 - alignment_value, 363
 - common_reference_t, 360
 - common_type_t, 360
 - conditional_t, 360
 - decay_t, 360
 - enable_if_t, 360
 - false_type, 360
 - is_corresponding_member, 362
 - is_layout_compatible_v, 363
 - is_pointer_interconvertible_base_of_v, 363
 - is_pointer_interconvertible_with_class, 362
 - make_signed_t, 361
 - make_unsigned_t, 361
 - remove_all_extents_t, 361
 - remove_extent_t, 361
 - remove_pointer_t, 361
 - remove_reference_t, 361
 - result_of_t, 361
 - swap, 362
 - true_type, 361
 - type, 361
 - underlying_type_t, 362
- microseconds
 - Time, 375
- milliseconds
 - Time, 375
- min
 - __gnu_parallel, 464
 - Numeric Arrays, 228
 - Sorting, 74, 75
 - std::bernoulli_distribution, 1770
 - std::binomial_distribution< _IntType >, 1800
 - std::cauchy_distribution< _RealType >, 1825
 - std::chi_squared_distribution< _RealType >, 1851
 - std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2093
 - std::discrete_distribution< _IntType >, 2097
 - std::exponential_distribution< _RealType >, 2158
 - std::extreme_value_distribution< _RealType >, 2162
 - std::fisher_f_distribution< _RealType >, 2172
 - std::gamma_distribution< _RealType >, 2218
 - std::geometric_distribution< _IntType >, 2222
 - std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2282
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2360
 - std::lognormal_distribution< _RealType >, 2407
 - std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2464
 - std::negative_binomial_distribution< _IntType >, 2562
 - std::normal_distribution< _RealType >, 2569
 - std::numeric_limits< _Tp >, 2608
 - std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2758
 - std::piecewise_constant_distribution< _RealType >, 2762
 - std::piecewise_linear_distribution< _RealType >, 2766
 - std::poisson_distribution< _IntType >, 2778
 - std::shuffle_order_engine< _RandomNumberEngine, __k >, 2926
 - std::student_t_distribution< _RealType >, 2988
 - std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2997
 - std::uniform_int_distribution< _IntType >, 3101
 - std::uniform_real_distribution< _RealType >, 3104
 - std::weibull_distribution< _RealType >, 3278
- min_element
 - Sorting, 75, 76
- min_element_minimal_n
 - __gnu_parallel::_Settings, 1041
- min_exponent

- std::numeric_limits_base, [817](#)
- std::numeric_limits< _Tp >, [2610](#)
- std::numeric_limits< bool >, [2614](#)
- std::numeric_limits< char >, [2618](#)
- std::numeric_limits< char16_t >, [2622](#)
- std::numeric_limits< char32_t >, [2626](#)
- std::numeric_limits< double >, [2630](#)
- std::numeric_limits< float >, [2634](#)
- std::numeric_limits< int >, [2638](#)
- std::numeric_limits< long >, [2642](#)
- std::numeric_limits< long double >, [2646](#)
- std::numeric_limits< long long >, [2650](#)
- std::numeric_limits< short >, [2654](#)
- std::numeric_limits< signed char >, [2658](#)
- std::numeric_limits< unsigned char >, [2662](#)
- std::numeric_limits< unsigned int >, [2666](#)
- std::numeric_limits< unsigned long >, [2670](#)
- std::numeric_limits< unsigned long long >, [2674](#)
- std::numeric_limits< unsigned short >, [2678](#)
- std::numeric_limits< wchar_t >, [2682](#)
- min_exponent10
 - std::numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2610](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- minmax
 - Sorting, [76](#), [77](#)
- minmax_element
 - Sorting, [77](#)
- minstd_rand
 - Random Number Generators, [249](#)
- minstd_rand0
 - Random Number Generators, [249](#)
- minutes
 - Time, [375](#)
- mismatch
 - Non-Mutating, [57](#), [58](#)
- mod_based_range_hashing.hpp, [3575](#)
- modulus
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, [2362](#)
- mofunc_impl.h, [3358](#)
- monetary
 - std::locale, [2397](#)
- money_get
 - std::money_get< _CharT, _InIter >, [2480](#)
- money_put
 - std::money_put< _CharT, _OutIter >, [2485](#)
- money_punct
 - std::money_punct< _CharT, _Intl >, [2490](#), [2491](#)
- monostate.h, [3358](#)
- months
 - Time, [375](#)
- move
 - Mutating, [28](#)
 - Utilities, [310](#)
- move.h, [3358](#)
- move_backward
 - Mutating, [29](#)
- move_if_noexcept
 - Utilities, [312](#)
- move_only_function
 - std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, [2508](#), [2509](#)
- mt19937
 - Random Number Generators, [249](#)
- mt19937_64
 - Random Number Generators, [250](#)
- mt_allocator.h, [3548](#)
- multiline
 - std::regex_constants, [730](#)
- multimap
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2517](#)–[2519](#)
- multimap.h, [3523](#)
- multiplier
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, [2362](#)
- multiseq_partition
 - __gnu_parallel, [464](#)
- multiseq_selection
 - __gnu_parallel, [464](#)
- multiseq_selection.h, [3670](#)
- multiset
 - std::multiset< _Key, _Compare, _Alloc >, [2541](#)–[2544](#)
- multiset.h, [3523](#)
- multiway_merge
 - __gnu_parallel, [465](#)
- multiway_merge.h, [3671](#)
- _GLIBCXX_PARALLEL_LENGTH, [3674](#)

- multiway_merge_3_variant
 - __gnu_parallel, 466
- multiway_merge_4_variant
 - __gnu_parallel, 467
- multiway_merge_exact_splitting
 - __gnu_parallel, 468
- multiway_merge_loser_tree
 - __gnu_parallel, 468
- multiway_merge_loser_tree_sentinel
 - __gnu_parallel, 468
- multiway_merge_loser_tree_unguarded
 - __gnu_parallel, 469
- multiway_merge_minimal_k
 - __gnu_parallel::Settings, 1041
- multiway_merge_minimal_n
 - __gnu_parallel::Settings, 1041
- multiway_merge_oversampling
 - __gnu_parallel::Settings, 1041
- multiway_merge_sampling_splitting
 - __gnu_parallel, 470
- multiway_merge_sentinels
 - __gnu_parallel, 470
- multiway_mergesort.h, 3674
- Mutating, 21
 - copy, 23
 - copy_backward, 24
 - copy_if, 24
 - copy_n, 25
 - fill, 25
 - fill_n, 26
 - generate, 26
 - generate_n, 27
 - is_partitioned, 27
 - iter_swap, 28
 - move, 28
 - move_backward, 29
 - partition, 29
 - partition_copy, 30
 - partition_point, 31
 - random_shuffle, 31
 - remove, 32
 - remove_copy, 32
 - remove_copy_if, 33
 - remove_if, 33
 - replace, 34
 - replace_copy_if, 34
 - replace_if, 35
 - reverse, 35
 - reverse_copy, 36
 - rotate, 36
 - rotate_copy, 37
 - shuffle, 38
 - stable_partition, 38
 - swap_ranges, 39
 - transform, 39, 40
 - unique, 41
 - unique_copy, 42
- mutex, 3645
- Mutexes, 122
 - adopt_lock, 124
 - call_once, 123
 - defer_lock, 124
 - lock, 123
 - try_lock, 124
 - try_to_lock, 125
- name
 - __gnu_debug::type_info, 3090
 - std::error_category, 2148
 - std::locale, 2394
 - std::type_info, 3091
- nanoseconds
 - Time, 376
- narrow
 - std::__ctype_abstract_base<_CharT>, 783, 784
 - std::basic_fstream<_CharT, _Traits>, 1162
 - std::basic_ifstream<_CharT, _Traits>, 1213
 - std::basic_ios<_CharT, _Traits>, 1248
 - std::basic_iostream<_CharT, _Traits>, 1278
 - std::basic_istream<_CharT, _Traits>, 1324
 - std::basic_istreamstream<_CharT, _Traits, _Alloc>, 1365
 - std::basic_ofstream<_CharT, _Traits>, 1400
 - std::basic_ostream<_CharT, _Traits>, 1432
 - std::basic_ostreamstream<_CharT, _Traits, _Alloc>, 1465
 - std::basic_stringstream<_CharT, _Traits, _Alloc>, 1740
 - std::ctype<_CharT>, 1960, 1961
 - std::ctype<char>, 1983–1985
 - std::ctype<wchar_t>, 2005
 - std::ctype_byname<_CharT>, 2019, 2020
 - std::ctype_byname<char>, 2039, 2040
- native_handle
 - std::thread, 3009
- neg_format
 - std::moneypunct<_CharT, _Intl>, 2495
 - std::moneypunct_byname<_CharT, _Intl>, 2502
- negative_sign
 - std::moneypunct<_CharT, _Intl>, 2495
 - std::moneypunct_byname<_CharT, _Intl>, 2503
- Negators, 325
 - not1, 326
 - not2, 326
- nested_exception
 - std::nested_exception, 2565
- nested_exception.h, 3359
- nested_ptr

- std::nested_exception, [2565](#)
- new, [3739](#)
 - operator new, [3740](#)
- new_allocator.h, [3360](#)
- new_delete_resource
 - Polymorphic memory resources, [355](#)
- new_except.h, [3360](#)
- new_handler
 - std, [591](#)
- new_throw.h, [3361](#)
- next_permutation
 - Sorting, [78](#)
- noboolalpha
 - std, [615](#)
- node.hpp, [3578](#)
- node_begin
 - __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2708](#)
 - __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, [2748](#)
 - __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2821](#)
 - __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2939](#)
- node_const_iterator
 - __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, [1782](#)
 - __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, [1782](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [3057](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [3059](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [3060](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [3061](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [3062](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [3063](#)
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3078](#)
 - __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3079](#)
- node_end
 - __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2708](#)
 - __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, [2749](#)
 - __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2821](#)
 - __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, [2939](#)
- node_handle.h, [3361](#)
- node_iterators.hpp, [3549](#), [3550](#)
- node_metadata_selector.hpp, [3588](#)
- node_type
 - __gnu_pbds::detail::pat_trie_base, [2746](#)
 - __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, [2748](#)
- node_update
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3078](#)
 - __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, [3079](#)
- Non-Mutating, [43](#)
 - adjacent_find, [45](#)
 - all_of, [46](#)
 - any_of, [46](#)
 - count, [47](#)
 - count_if, [47](#)
 - equal, [48](#), [49](#)
 - find, [50](#)
 - find_end, [50](#), [51](#)
 - find_first_of, [52](#)
 - find_if, [53](#)
 - find_if_not, [53](#)
 - for_each, [54](#)
 - for_each_n, [54](#)
 - is_permutation, [55](#), [56](#)
 - mismatch, [57](#), [58](#)
 - none_of, [59](#)
 - search, [59](#), [60](#)
 - search_n, [61](#)
- none
 - std::bitset< _Nb >, [1816](#)
 - std::locale, [2397](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2111](#)
- none_of
 - Non-Mutating, [59](#)
- norm
 - Complex Numbers, [187](#)
 - std::tr1, [739](#)
- Normal Distributions, [241](#)
- operator<=, [241](#)

- operator>>, 242
- normal_distribution
 - std::normal_distribution< _RealType >, 2569
- noshowbase
 - std, 615
- noshowpoint
 - std, 615
- noshowpos
 - std, 616
- noskipws
 - std, 616
- nosubs
 - std::regex_constants, 731
- not1
 - Negators, 326
- not2
 - Negators, 326
- not_fn
 - std::experimental, 715
- notify_cleared
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1828
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2263
 - __gnu_pbds::sample_resize_policy, 2868
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_erase_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1828
 - __gnu_pbds::sample_resize_policy, 2868
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_erase_search_end
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1828
 - __gnu_pbds::sample_resize_policy, 2868
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_erase_search_start
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1828
 - __gnu_pbds::sample_resize_policy, 2868
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_erased
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2868
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_externally_resized
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_find_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2868
- __gnu_pbds::sample_resize_trigger, 2871
- notify_find_search_end
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_find_search_start
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2871
- notify_insert_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2872
- notify_insert_search_end
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1829
 - __gnu_pbds::sample_resize_policy, 2869
- notify_insert_search_start
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1830
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2872
- notify_inserted
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1830
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2263
 - __gnu_pbds::sample_resize_policy, 2869
- notify_resized
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1830
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2263
 - __gnu_pbds::sample_range_hashing, 2865
 - __gnu_pbds::sample_ranged_hash_fn, 2866
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2872
- __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 887
- __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1589
- nth_element
 - Sorting, 79

- nth_element_minimal_n
 - __gnu_parallel::Settings, 1041
- null_memory_resource
 - memory_resource, 3644
- null_node_metadata.hpp, 3567
- nullopt
 - Optional values, 294
- num_blocks
 - std::tr2::dynamic_bitset<_WordT, _Alloc>, 2111
- num_children
 - __gnu_pbds::detail::pat_trie_base::_Node_citer<Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc>, 965
 - __gnu_pbds::detail::pat_trie_base::_Node_iter<Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc>, 968
- num_get
 - std::num_get<_CharT, _InIter>, 2578
- num_put
 - std::num_put<_CharT, _OutIter>, 2595
- numbers, 3645
- numeric, 3645–3647
 - std::locale, 2397
- Numeric Arrays, 221
 - ~gslice, 226
 - apply, 226, 227
 - cshift, 227
 - gslice, 223
 - gslice_array, 224
 - indirect_array, 224
 - mask_array, 224
 - max, 228
 - min, 228
 - operator=, 228–231
 - operator[], 232–235
 - resize, 235
 - shift, 236
 - size, 236
 - slice, 224, 225
 - slice_array, 225
 - start, 236, 237
 - stride, 237
 - sum, 237
 - swap, 237
 - valarray, 225, 226
- numeric_traits.h, 3540
- numeric_fwd.h, 3674
- Numerics, 181
- num_punct
 - std::num_punct<_CharT>, 2685
- oct
 - std, 616
 - std::basic_fstream<_CharT, _Traits>, 1190
 - std::basic_ifstream<_CharT, _Traits>, 1233
 - std::basic_ios<_CharT, _Traits>, 1257
 - std::basic_iostream<_CharT, _Traits>, 1304
 - std::basic_istream<_CharT, _Traits>, 1344
 - std::basic_istreamstream<_CharT, _Traits, _Alloc>, 1384
 - std::basic_ofstream<_CharT, _Traits>, 1418
 - std::basic_ostream<_CharT, _Traits>, 1449
 - std::basic_ostringstream<_CharT, _Traits, _Alloc>, 1483
 - std::basic_stringstream<_CharT, _Traits, _Alloc>, 1767
 - std::ios_base, 2308
- off_type
 - std::basic_ios<_CharT, _Traits>, 1241
 - std::basic_streambuf<_CharT, _Traits>, 1497
 - std::wbuffer_convert<_Codecvt, _Elem, _Tr>, 3261
- ofstream
 - I/O, 170
- omp_loop.h, 3676
- omp_loop_static.h, 3676
- once_flag
 - std::once_flag, 2695
- open
 - __gnu_cxx::enc_filebuf<_CharT>, 2125
 - __gnu_cxx::stdio_filebuf<_CharT, _Traits>, 2952, 2953
 - std::basic_filebuf<_CharT, _Traits>, 1127, 1128
 - std::basic_fstream<_CharT, _Traits>, 1163
 - std::basic_ifstream<_CharT, _Traits>, 1213, 1214
 - std::basic_ofstream<_CharT, _Traits>, 1400, 1401
- openmode
 - std::basic_fstream<_CharT, _Traits>, 1149
 - std::basic_ifstream<_CharT, _Traits>, 1200
 - std::basic_ios<_CharT, _Traits>, 1241
 - std::basic_iostream<_CharT, _Traits>, 1266
 - std::basic_istream<_CharT, _Traits>, 1313
 - std::basic_istreamstream<_CharT, _Traits, _Alloc>, 1353
 - std::basic_ofstream<_CharT, _Traits>, 1392
 - std::basic_ostream<_CharT, _Traits>, 1426
 - std::basic_ostringstream<_CharT, _Traits, _Alloc>, 1458
 - std::basic_stringstream<_CharT, _Traits, _Alloc>, 1728
 - std::ios_base, 2299
- operator_iterator
 - __gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category>, 997
 - __gnu_debug::Safe_local_iterator<_Iterator, _UContainer>, 1011
- operator_RAlter
 - __gnu_parallel::GuardedIterator<_RAIter, _Compare>, 916

operator bool

[std::basic_fstream< _CharT, _Traits >, 1164](#)
[std::basic_ifstream< _CharT, _Traits >, 1214](#)
[std::basic_ios< _CharT, _Traits >, 1248](#)
[std::basic_iostream< _CharT, _Traits >, 1278](#)
[std::basic_istream< _CharT, _Traits >, 1325](#)
[std::basic_istream< _CharT, _Traits >::sentry, 2883](#)
[std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1365](#)
[std::basic_ofstream< _CharT, _Traits >, 1401](#)
[std::basic_ostream< _CharT, _Traits >, 1433](#)
[std::basic_ostream< _CharT, _Traits >::sentry, 2885](#)
[std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1466](#)
[std::basic_stringstream< _CharT, _Traits, _Alloc >, 1740](#)
[std::copyable_function< _Res\(_ArgTypes...\) _GLIBCXX_MOF_CV noexcept\(_Noex\)>, 1949](#)
[std::error_code, 2150](#)
[std::error_condition, 2152](#)
[std::function< _Res\(_ArgTypes...\)>, 2205](#)
[std::move_only_function< _Res\(_ArgTypes...\) _GLIBCXX_MOF_CV noexcept\(_Noex\)>, 2510](#)
[std::shared_ptr< _Tp >, 2922](#)
[std::tr2::bool_set, 1822](#)
[std::unique_ptr< _Tp, _Dp >, 3110](#)
[std::unique_ptr< _Tp\[\], _Dp >, 3115](#)

operator new

[new, 3740](#)

operator streamoff

[std::fpos< _StateT >, 2197](#)

operator string_type

[std::sub_match< _Bilter >, 2993](#)

operator!

[std::basic_fstream< _CharT, _Traits >, 1164](#)
[std::basic_ifstream< _CharT, _Traits >, 1214](#)
[std::basic_ios< _CharT, _Traits >, 1248](#)
[std::basic_iostream< _CharT, _Traits >, 1278](#)
[std::basic_istream< _CharT, _Traits >, 1325](#)
[std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1365](#)
[std::basic_ofstream< _CharT, _Traits >, 1401](#)
[std::basic_ostream< _CharT, _Traits >, 1433](#)
[std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1466](#)
[std::basic_stringstream< _CharT, _Traits, _Alloc >, 1741](#)
[std::valarray< _Tp >, 3227](#)

operator!=

[__gnu_cxx, 405](#)
[__gnu_pbds::detail::bin_search_tree_const_node_iterator< Node, Const_iterator, Iterator, _Alloc >, 1776](#)
[__gnu_pbds::detail::bin_search_tree_node_iterator< Node, Const_iterator, Iterator, _Alloc >, 1780](#)

[__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1790](#)
[__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1794](#)
[__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2346](#)
[__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2350](#)
[__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, 965](#)
[__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, 968](#)
[Dynamic Bitset., 140](#)
[std, 616, 617](#)

std::ops, 731

operator<

[__gnu_cxx, 408](#)
[__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 916](#)
[Pointer Abstractions, 349, 350](#)
[std, 619–621](#)
[Time, 384](#)

operator<<

[Bernoulli Distributions, 239, 240](#)
[Complex Numbers, 189](#)
[Diagnostics, 131](#)
[Dynamic Bitset., 141](#)
[Normal Distributions, 241](#)
[Pointer Abstractions, 350](#)
[Poisson Distributions, 243, 244](#)
[Random Number Generators, 250](#)
[Regular Expressions, 261](#)
[std, 621–627](#)
[std::_detail, 668, 669](#)
[std::basic_fstream< _CharT, _Traits >, 1164–1169](#)
[std::basic_iostream< _CharT, _Traits >, 1278–1283](#)
[std::basic_ofstream< _CharT, _Traits >, 1402–1407](#)
[std::basic_ostream< _CharT, _Traits >, 1433–1438](#)
[std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1466–1471](#)
[std::basic_stringstream< _CharT, _Traits, _Alloc >, 1741–1746](#)
[std::binomial_distribution< _IntType >, 1801](#)
[std::bitset< _Nb >, 1816](#)
[std::chi_squared_distribution< _RealType >, 1852](#)
[std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2094](#)
[std::discrete_distribution< _IntType >, 2098](#)
[std::filesystem::path, 2756](#)
[std::fisher_f_distribution< _RealType >, 2173](#)
[std::gamma_distribution< _RealType >, 2219](#)
[std::linear_congruential_engine< _UIntType, __a,](#)

- `__c, __m` >, 2361
 - `std::lognormal_distribution< _RealType` >, 2408
 - `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f` >, 2465
 - `std::negative_binomial_distribution< _IntType` >, 2563
 - `std::normal_distribution< _RealType` >, 2570
 - `std::philox_engine< _UIntType, __w, __n, __r, __consts` >, 2759
 - `std::piecewise_constant_distribution< _RealType` >, 2763
 - `std::piecewise_linear_distribution< _RealType` >, 2767
 - `std::poisson_distribution< _IntType` >, 2779
 - `std::shuffle_order_engine< _RandomNumberEngine, __k` >, 2927
 - `std::student_t_distribution< _RealType` >, 2989
 - `std::subtract_with_carry_engine< _UIntType, __w, __s, __r` >, 2998
 - `std::tr2::dynamic_bitset< _WordT, _Alloc` >, 2112
 - `std::unique_ptr< _Tp[], _Dp` >, 3117
 - Time, 384
 - Uniform Distributions, 246, 247
- operator<=<=
 - `std::bitset< _Nb` >, 1816
 - `std::gslice_array< _Tp` >, 2243
 - `std::indirect_array< _Tp` >, 2285
 - `std::mask_array< _Tp` >, 2443
 - `std::slice_array< _Tp` >, 2932
 - `std::tr2::dynamic_bitset< _WordT, _Alloc` >, 2112
 - `std::valarray< _Tp` >, 3229
- operator<=
 - `__gnu_cxx`, 409, 410
 - `__gnu_parallel::GuardedIterator< _RAIter, _Compare` >, 916
 - Dynamic Bitset., 141
 - Pointer Abstractions, 350
 - std, 627, 628
 - `std::rel_ops`, 732
 - Time, 384
- operator<=>
 - `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory` >, 927
 - Diagnostics, 131
 - Regular Expressions, 261, 262
 - std, 628–631
 - `std::error_category`, 2149
 - `std::filesystem::path`, 2756
 - Time, 385
 - Utilities, 312
- operator>
 - `__gnu_cxx`, 411, 412
 - Dynamic Bitset., 141
 - Pointer Abstractions, 351
 - std, 636, 637
 - `std::rel_ops`, 732
 - Time, 385
- operator>>
 - Bernoulli Distributions, 240
 - Complex Numbers, 189
 - Dynamic Bitset., 142
 - Normal Distributions, 242
 - Poisson Distributions, 244, 245
 - std, 638–643
 - `std::__detail`, 669
 - `std::basic_fstream< _CharT, _Traits` >, 1169–1174
 - `std::basic_ifstream< _CharT, _Traits` >, 1215–1220
 - `std::basic_iostream< _CharT, _Traits` >, 1284–1289
 - `std::basic_istream< _CharT, _Traits` >, 1325–1330
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc` >, 1366–1371
 - `std::basic_stringstream< _CharT, _Traits, _Alloc` >, 1746–1751
 - `std::binomial_distribution< _IntType` >, 1802
 - `std::bitset< _Nb` >, 1817
 - `std::chi_squared_distribution< _RealType` >, 1852
 - `std::discard_block_engine< _RandomNumberEngine, __p, __r` >, 2095
 - `std::discrete_distribution< _IntType` >, 2098
 - `std::filesystem::path`, 2756
 - `std::fisher_f_distribution< _RealType` >, 2173
 - `std::gamma_distribution< _RealType` >, 2220
 - `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType` >, 2283
 - `std::linear_congruential_engine< _UIntType, __a, __c, __m` >, 2361
 - `std::lognormal_distribution< _RealType` >, 2408
 - `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f` >, 2466
 - `std::negative_binomial_distribution< _IntType` >, 2564
 - `std::normal_distribution< _RealType` >, 2571
 - `std::philox_engine< _UIntType, __w, __n, __r, __consts` >, 2760
 - `std::piecewise_constant_distribution< _RealType` >, 2763
 - `std::piecewise_linear_distribution< _RealType` >, 2767
 - `std::poisson_distribution< _IntType` >, 2779
 - `std::shuffle_order_engine< _RandomNumberEngine, __k` >, 2928
 - `std::student_t_distribution< _RealType` >, 2990
 - `std::subtract_with_carry_engine< _UIntType, __w, __s, __r` >, 2998
 - `std::tr2::dynamic_bitset< _WordT, _Alloc` >, 2113
 - Uniform Distributions, 247

operator>>=
 std::bitset< _Nb >, 1817
 std::gslice_array< _Tp >, 2243
 std::indirect_array< _Tp >, 2286
 std::mask_array< _Tp >, 2443
 std::slice_array< _Tp >, 2932
 std::tr2::dynamic_bitset< _WordT, _Alloc >, 2113
 std::valarray< _Tp >, 3229

operator>=
 __gnu_cxx, 412, 413
 Dynamic Bitset., 142
 Pointer Abstractions, 352
 std, 637, 638
 std::rel_ops, 732
 Time, 385

operator()
 __gnu_cxx::subtractive_rng, 3000
 __gnu_parallel::_Nothing, 971
 __gnu_parallel::_RandomNumber, 982
 __gnu_parallel::_accumulate_selector< _It >, 753
 __gnu_parallel::_adjacent_find_selector, 756
 __gnu_parallel::_count_if_selector< _It, _Diff >, 772
 __gnu_parallel::_count_selector< _It, _Diff >, 774
 __gnu_parallel::_fill_selector< _It >, 791
 __gnu_parallel::_find_first_of_selector< _FIterator >, 792
 __gnu_parallel::_find_if_selector, 794
 __gnu_parallel::_for_each_selector< _It >, 795
 __gnu_parallel::_generate_selector< _It >, 797
 __gnu_parallel::_identity_selector< _It >, 801
 __gnu_parallel::_inner_product_selector< _It, _It2, _Tp >, 803
 __gnu_parallel::_mismatch_selector, 807
 __gnu_parallel::_replace_if_selector< _It, _Op, _Tp >, 828
 __gnu_parallel::_replace_selector< _It, _Tp >, 830
 __gnu_parallel::_transform1_selector< _It >, 831
 __gnu_parallel::_transform2_selector< _It >, 833
 __gnu_pbds::direct_mask_range_hashing< Size_Type >, 2087
 __gnu_pbds::direct_mod_range_hashing< Size_Type >, 2088
 __gnu_pbds::linear_probe_fn< Size_Type >, 2363
 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2411
 __gnu_pbds::lu_move_to_front_policy< _Alloc >, 2415
 __gnu_pbds::quadratic_probe_fn< Size_Type >, 2793
 __gnu_pbds::sample_probe_fn, 2864
 __gnu_pbds::sample_range_hashing, 2865
 __gnu_pbds::sample_ranged_hash_fn, 2866
 __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 2875
 __gnu_pbds::sample_update_policy, 2876
 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >, 3055
 __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3069
 __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3073
 std::bernoulli_distribution, 1770
 std::binomial_distribution< _IntType >, 1800
 std::cauchy_distribution< _RealType >, 1825
 std::chi_squared_distribution< _RealType >, 1851
 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1949
 std::default_delete< _Tp >, 2055
 std::default_delete< _Tp[]>, 2056
 std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2094
 std::discrete_distribution< _IntType >, 2097
 std::exponential_distribution< _RealType >, 2158
 std::extreme_value_distribution< _RealType >, 2162
 std::fisher_f_distribution< _RealType >, 2172
 std::function< _Res(_ArgTypes...)>, 2205
 std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2210
 std::gamma_distribution< _RealType >, 2218, 2219
 std::geometric_distribution< _IntType >, 2222
 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2282
 std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2360
 std::locale, 2394
 std::lognormal_distribution< _RealType >, 2407
 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2510
 std::negative_binomial_distribution< _IntType >, 2562
 std::normal_distribution< _RealType >, 2569
 std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2759
 std::piecewise_constant_distribution< _RealType >, 2762
 std::piecewise_linear_distribution< _RealType >, 2766
 std::plus< _Tp >, 2769
 std::poisson_distribution< _IntType >, 2778
 std::shuffle_order_engine< _RandomNumberEngine, __k >, 2926
 std::student_t_distribution< _RealType >, 2988
 std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2997
 std::uniform_int_distribution< _IntType >, 3101
 std::uniform_real_distribution< _RealType >, 3104

- std::weibull_distribution< _RealType >, 3278
- operator+
 - __gnu_cxx, 406, 407
 - Complex Numbers, 187, 188
 - std, 617–619
 - std::fpos< _StateT >, 2197
 - std::reverse_iterator< _Iterator >, 2855
 - std::valarray< _Tp >, 3228
 - Time, 382, 383
- operator++
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, 997, 998
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, 1012
 - __gnu_parallel::GuardedIterator< _RAIter, _Compare >, 916
 - std::back_insert_iterator< _Container >, 1107
 - std::front_insert_iterator< _Container >, 2201, 2202
 - std::insert_iterator< _Container >, 2291
 - std::istreambuf_iterator< _CharT, _Traits >, 2335
 - std::ostreambuf_iterator< _CharT, _Traits >, 2703
 - std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2837
 - std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2841
 - std::reverse_iterator< _Iterator >, 2855
- operator+=
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 871, 872
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1568
 - std::basic_string< _CharT, _Traits, _Alloc >, 1662–1665
 - std::complex< _Tp >, 1913
 - std::fpos< _StateT >, 2197
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::reverse_iterator< _Iterator >, 2856
 - std::slice_array< _Tp >, 2932
 - std::valarray< _Tp >, 3228
- operator-
 - Complex Numbers, 188
 - Dynamic Bitset., 141
 - std::fpos< _StateT >, 2197
 - std::reverse_iterator< _Iterator >, 2856
 - std::valarray< _Tp >, 3228
 - Time, 383
- operator->
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, 998
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, 1012
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1791
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1794
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2346
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2350
 - std::auto_ptr< _Tp >, 1103
 - std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2837
 - std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2841
 - std::reverse_iterator< _Iterator >, 2857
 - std::unique_ptr< _Tp, _Dp >, 3110
 - std::unique_ptr< _Tp[], _Dp >, 3116
- operator--
 - std::reverse_iterator< _Iterator >, 2856
- operator-=
 - std::complex< _Tp >, 1913
 - std::fpos< _StateT >, 2197
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::reverse_iterator< _Iterator >, 2856
 - std::slice_array< _Tp >, 2932
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2112
 - std::valarray< _Tp >, 3228, 3229
- operator/
 - Complex Numbers, 188, 189
 - std::filesystem::path, 2756
 - Time, 383, 384
- operator/=
 - std::complex< _Tp >, 1914
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::slice_array< _Tp >, 2932
 - std::valarray< _Tp >, 3229
- operator=
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 873, 874
 - __gnu_cxx::pair< _T1, _T2 >, 2723, 2724
 - __gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >, 998
 - __gnu_debug::Safe_local_iterator< _Iterator, _UContainer >, 1012
 - Numeric Arrays, 228–231
 - std::auto_ptr< _Tp >, 1103, 1104
 - std::back_insert_iterator< _Container >, 1108
 - std::basic_regex< _Ch_type, _Rx_traits >, 1492, 1493
 - std::basic_string< _CharT, _Traits, _Alloc >, 1665–1667
 - std::complex< _Tp >, 1914

std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CWNode, _Alloc >, 2350
 noexcept(_Noex)>, 1949
 std::deque< _Tp, _Alloc >, 2081, 2082
 std::experimental::fundamentals_v1::any, 1085, 1086
 std::forward_list< _Tp, _Alloc >, 2190, 2191
 std::front_insert_iterator< _Container >, 2202
 std::function< _Res(_ArgTypes...) >, 2205–2207
 std::insert_iterator< _Container >, 2292
 std::list< _Tp, _Alloc >, 2380
 std::locale, 2394
 std::map< _Key, _Tp, _Compare, _Alloc >, 2438, 2439
 std::match_results< _Bi_iter, _Alloc >, 2451
 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2510
 std::multimap< _Key, _Tp, _Compare, _Alloc >, 2531, 2532
 std::multiset< _Key, _Compare, _Alloc >, 2553
 std::once_flag, 2695
 std::ostream_iterator< _Tp, _CharT, _Traits >, 2700
 std::ostreambuf_iterator< _CharT, _Traits >, 2704
 std::pair< _T1, _T2 >, 2728, 2729
 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2837
 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2842
 std::set< _Key, _Compare, _Alloc >, 2905
 std::tr2::dynamic_bitset< _WordT, _Alloc >, 2113
 std::unique_ptr< _Tp, _Dp >, 3110, 3111
 std::unique_ptr< _Tp[], _Dp >, 3116
 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3144
 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3195
 std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3220
 std::vector< _Tp, _Alloc >, 3248, 3249
 operator==
 __gnu_cxx, 410, 411
 __gnu_parallel::iterator_pair< _Iterator1, _Iterator2, _IteratorCategory >, 927
 __gnu_pbds::detail::bin_search_tree_const_node_iterator< Node, Const_Iterator, Iterator, _Alloc >, 1776
 __gnu_pbds::detail::bin_search_tree_node_iterator< Node, Const_Iterator, Iterator, _Alloc >, 1781
 __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1791
 __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1794
 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2346
 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2346
 __gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc >, 966
 __gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc >, 969
 Allocators, 334
 Complex Numbers, 189
 Diagnostics, 132
 Iterators, 177
 Pointer Abstractions, 350, 351
 Regular Expressions, 262–264
 std, 631–636
 std::_fwdlist::iterator< _Const, _Ptr >, 922
 std::_Fwd_list_const_iterator< _Tp >, 912
 std::_Fwd_list_iterator< _Tp >, 913
 std::allocator< void >, 1060
 std::bernoulli_distribution, 1771
 std::binomial_distribution< _IntType >, 1802
 std::bitset< _Nb >, 1817
 std::cauchy_distribution< _RealType >, 1826
 std::chi_squared_distribution< _RealType >, 1852
 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1950
 std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2095
 std::discrete_distribution< _IntType >, 2098
 std::error_category, 2149
 std::exponential_distribution< _RealType >, 2159
 std::extreme_value_distribution< _RealType >, 2163
 std::filesystem::path, 2756
 std::fisher_f_distribution< _RealType >, 2173
 std::gamma_distribution< _RealType >, 2220
 std::geometric_distribution< _IntType >, 2223
 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2283
 std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2332
 std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2361
 std::locale, 2395
 std::lognormal_distribution< _RealType >, 2408
 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2465
 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2511
 std::negative_binomial_distribution< _IntType >, 2564
 std::normal_distribution< _RealType >, 2571
 std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2760
 std::poisson_distribution< _RealType >, 2760

- 2763
- std::piecewise_linear_distribution< _RealType >, 2767
- std::poisson_distribution< _IntType >, 2779
- std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2837
- std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2842
- std::shuffle_order_engine< _RandomNumberEngine, __k >, 2927
- std::slice, 2930
- std::student_t_distribution< _RealType >, 2989
- std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2998
- std::uniform_int_distribution< _IntType >, 3102
- std::uniform_real_distribution< _RealType >, 3105
- std::weibull_distribution< _RealType >, 3279
- Time, 385
- Utilities, 312, 313
- operator%
 - Time, 382
- operator%=
 - std::gslice_array< _Tp >, 2242
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::slice_array< _Tp >, 2932
 - std::valarray< _Tp >, 3227
- operator&
 - Dynamic Bitset., 140
 - std, 617
 - std::regex_constants, 725
- operator&=
 - std::bitset< _Nb >, 1816
 - std::gslice_array< _Tp >, 2242
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::regex_constants, 725
 - std::slice_array< _Tp >, 2932
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2111, 2112
 - std::valarray< _Tp >, 3227
- operator[]
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __Base >, 874
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1568, 1569
 - Numeric Arrays, 232–235
 - std::basic_string< _CharT, _Traits, _Alloc >, 1668, 1669
 - std::bitset< _Nb >, 1817
 - std::deque< _Tp, _Alloc >, 2082
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2439
 - std::match_results< _Bi_iter, _Alloc >, 2451
 - std::reverse_iterator< _Iterator >, 2857
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2113, 2114
 - std::unique_ptr< _Tp[], _Dp >, 3116
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3144, 3145
 - std::vector< _Tp, _Alloc >, 3249
- operator*
 - __gnu_debug::Safe_iterator< _Iterator, __Sequence, __Category >, 997
 - __gnu_debug::Safe_local_iterator< _Iterator, __UContainer >, 1011
 - __gnu_parallel::GuardedIterator< _RAIter, __Compare >, 916
 - __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1776
 - __gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1781
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1791
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1794
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2346
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2350
 - __gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >, 2712
 - __gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, __Cliterator, Iterator, _Alloc >, 966
 - __gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, __Cliterator, Iterator, _Alloc >, 969
 - Complex Numbers, 187
 - std::auto_ptr< _Tp >, 1103
 - std::back_insert_iterator< _Container >, 1107
 - std::front_insert_iterator< _Container >, 2201
 - std::insert_iterator< _Container >, 2291
 - std::istreambuf_iterator< _CharT, _Traits >, 2335
 - std::ostreambuf_iterator< _CharT, _Traits >, 2703
 - std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2837
 - std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2841
 - std::reverse_iterator< _Iterator >, 2855
 - std::unique_ptr< _Tp, _Dp >, 3110
 - std::unique_ptr< _Tp[], _Dp >, 3115
 - Time, 382
- operator*=
 - std::complex< _Tp >, 1913
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2285
 - std::mask_array< _Tp >, 2443
 - std::slice_array< _Tp >, 2932

- std::valarray< _Tp >, 3228
- operator~
 - std::bitset< _Nb >, 1818
 - std::regex_constants, 727
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2114
 - std::valarray< _Tp >, 3230
- operator^
 - Dynamic Bitset., 142
 - std, 643
 - std::regex_constants, 726
- operator^=
 - std::bitset< _Nb >, 1818
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2286
 - std::mask_array< _Tp >, 2444
 - std::regex_constants, 726
 - std::slice_array< _Tp >, 2933
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2114
 - std::valarray< _Tp >, 3230
- operator|
 - Dynamic Bitset., 143
 - std, 643
 - std::regex_constants, 726
- operator|=
 - std::bitset< _Nb >, 1818
 - std::gslice_array< _Tp >, 2243
 - std::indirect_array< _Tp >, 2286
 - std::mask_array< _Tp >, 2444
 - std::regex_constants, 727
 - std::slice_array< _Tp >, 2933
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2114
 - std::valarray< _Tp >, 3230
- opt_random.h, 3734
- optimize
 - std::regex_constants, 731
- optional, 3649, 3650
- Optional values, 293
 - in_place, 294
 - nullopt, 294
- order_of_key
 - __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >, 3055
 - __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3069
- order_of_prefix
 - __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 3069
- order_preserving
 - __gnu_pbds::container_traits< Cntnr >, 1941
- order_statistics_imp.hpp, 3588, 3589
- os_defines.h, 3734
- ostream, 3650
 - I/O, 170
- ostream.h, 3361
- ostream.tcc, 3362
- ostream_insert.h, 3363
- ostream_iterator
 - std::ostream_iterator< _Tp, _CharT, _Traits >, 2699, 2700
- ostream_print.h, 3363
- ostream_type
 - std::ostream_iterator< _Tp, _CharT, _Traits >, 2698
 - std::ostreambuf_iterator< _CharT, _Traits >, 2702
- ostreambuf_iterator
 - std::ostreambuf_iterator< _CharT, _Traits >, 2703
- ostringstream
 - I/O, 171
- out
 - std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 770
 - std::basic_fstream< _CharT, _Traits >, 1190
 - std::basic_ifstream< _CharT, _Traits >, 1233
 - std::basic_ios< _CharT, _Traits >, 1258
 - std::basic_iostream< _CharT, _Traits >, 1305
 - std::basic_istream< _CharT, _Traits >, 1344
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1385
 - std::basic_ofstream< _CharT, _Traits >, 1418
 - std::basic_ostream< _CharT, _Traits >, 1450
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1483
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1767
 - std::codecvt< _InternT, _ExternT, _StateT >, 1856
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1863
 - std::codecvt< char, char, mbstate_t >, 1870, 1871
 - std::codecvt< char16_t, char, mbstate_t >, 1878
 - std::codecvt< char32_t, char, mbstate_t >, 1885, 1886
 - std::codecvt< wchar_t, char, mbstate_t >, 1893
 - std::codecvt_byname< _InternT, _ExternT, _StateT >, 1900
 - std::ios_base, 2308
- out_ptr
 - std, 644
- out_ptr.h, 3363
- ov_tree_map.hpp, 3580
- overflow
 - __gnu_cxx::enc_filebuf< _CharT >, 2126
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2954
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2971
 - std::basic_filebuf< _CharT, _Traits >, 1128
 - std::basic_streambuf< _CharT, _Traits >, 1501
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1708
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3264
- owner_before

- std::shared_ptr< _Tp >, [2922](#)
- p
 - std::bernoulli_distribution, [1770](#)
 - std::binomial_distribution< _IntType >, [1801](#)
 - std::geometric_distribution< _IntType >, [2222](#)
 - std::negative_binomial_distribution< _IntType >, [2563](#)
- pair
 - __gnu_cxx::pair< _T1, _T2 >, [2721](#)
 - std::pair< _T1, _T2 >, [2727](#)
 - Utilities, [305](#)
- pairing_heap.hpp, [3581](#)
- par_loop.h, [3677](#)
- parallel.h, [3677](#)
- parallel_balanced
 - __gnu_parallel, [432](#)
- parallel_multiway_merge
 - __gnu_parallel, [472](#)
- parallel_omp_loop
 - __gnu_parallel, [432](#)
- parallel_omp_loop_static
 - __gnu_parallel, [432](#)
- parallel_sort_mwms
 - __gnu_parallel, [472](#)
- parallel_sort_mwms_pu
 - __gnu_parallel, [473](#)
- parallel_tag
 - __gnu_parallel::parallel_tag, [2734](#)
- parallel_taskqueue
 - __gnu_parallel, [432](#)
- parallel_unbalanced
 - __gnu_parallel, [432](#)
- Parallelism TS, [301](#)
- parallelism_v2::simd_abi::deduce< _Tp, _Np,... >, [2053](#)
- param
 - std::bernoulli_distribution, [1770](#)
 - std::binomial_distribution< _IntType >, [1801](#)
 - std::cauchy_distribution< _RealType >, [1826](#)
 - std::chi_squared_distribution< _RealType >, [1851](#)
 - std::discrete_distribution< _IntType >, [2097](#)
 - std::exponential_distribution< _RealType >, [2158](#), [2159](#)
 - std::extreme_value_distribution< _RealType >, [2162](#)
 - std::fisher_f_distribution< _RealType >, [2172](#)
 - std::gamma_distribution< _RealType >, [2219](#)
 - std::geometric_distribution< _IntType >, [2222](#)
 - std::lognormal_distribution< _RealType >, [2407](#)
 - std::negative_binomial_distribution< _IntType >, [2563](#)
 - std::normal_distribution< _RealType >, [2570](#)
 - std::piecewise_constant_distribution< _RealType >, [2762](#), [2763](#)
 - std::piecewise_linear_distribution< _RealType >, [2766](#)
 - std::poisson_distribution< _IntType >, [2778](#)
 - std::student_t_distribution< _RealType >, [2989](#)
 - std::uniform_int_distribution< _IntType >, [3102](#)
 - std::uniform_real_distribution< _RealType >, [3104](#)
 - std::weibull_distribution< _RealType >, [3278](#)
- parse_numbers.h, [3364](#)
- partial_sort
 - Sorting, [80](#)
- partial_sort_copy
 - Sorting, [81](#)
- partial_sort_minimal_n
 - __gnu_parallel::Settings, [1042](#)
- partial_sum
 - Generalized Numeric operations, [14](#)
- partial_sum.h, [3677](#)
- partial_sum_dilation
 - __gnu_parallel::Settings, [1042](#)
- partial_sum_minimal_n
 - __gnu_parallel::Settings, [1042](#)
- partition
 - Mutating, [29](#)
- partition.h, [3678](#)
 - _GLIBCXX_VOLATILE, [3678](#)
- partition_chunk_share
 - __gnu_parallel::Settings, [1042](#)
- partition_chunk_size
 - __gnu_parallel::Settings, [1042](#)
- partition_copy
 - Mutating, [30](#)
- partition_minimal_n
 - __gnu_parallel::Settings, [1042](#)
- partition_point
 - Mutating, [31](#)
- pat_trie.hpp, [3581](#)
- pat_trie_base.hpp, [3582](#)
- pbackfail
 - __gnu_cxx::enc_filebuf< _CharT >, [2127](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2954](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2972](#)
 - std::basic_filebuf< _CharT, _Traits >, [1129](#)
 - std::basic_streambuf< _CharT, _Traits >, [1501](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1708](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3265](#)
- pbase
 - __gnu_cxx::enc_filebuf< _CharT >, [2127](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2955](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2972](#)
 - std::basic_filebuf< _CharT, _Traits >, [1129](#)
 - std::basic_streambuf< _CharT, _Traits >, [1502](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1709](#)

- std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3265](#)
- pbump
 - __gnu_cxx::enc_filebuf< _CharT >, [2127](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2955](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2973](#)
 - std::basic_filebuf< _CharT, _Traits >, [1130](#)
 - std::basic_streambuf< _CharT, _Traits >, [1502](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1709](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3265](#)
- peek
 - std::basic_fstream< _CharT, _Traits >, [1175](#)
 - std::basic_ifstream< _CharT, _Traits >, [1220](#)
 - std::basic_iostream< _CharT, _Traits >, [1289](#)
 - std::basic_istream< _CharT, _Traits >, [1331](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1371](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1752](#)
- perm_options
 - Filesystem, [167](#)
- perms
 - Filesystem, [167](#)
 - Filesystem TS, [286](#)
- philox4x32
 - Random Number Generators, [250](#)
- philox4x64
 - Random Number Generators, [250](#)
- philox_engine
 - std::philox_engine< _UIntType, __w, __n, __r, __consts >, [2758](#)
- piecewise_construct
 - Utilities, [314](#)
- pod_char_traits.h, [3595](#)
- point_const_iterator.hpp, [3563](#)
- point_iterator.hpp, [3591](#)
- point_iterators.hpp, [3550](#)
- pointer
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, [1790](#)
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, [1793](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, [2345](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_iterator< Node, _Alloc >, [2349](#)
 - std::allocator_traits< _Alloc >, [1064](#)
 - std::allocator_traits< allocator< _Tp > >, [1069](#)
 - std::allocator_traits< allocator< void > >, [1074](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1078](#)
 - std::back_insert_iterator< _Container >, [1107](#)
 - std::front_insert_iterator< _Container >, [2201](#)
 - std::insert_iterator< _Container >, [2291](#)
 - std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, [2331](#)
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
 - std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, [2340](#)
 - std::ostream_iterator< _Tp, _CharT, _Traits >, [2699](#)
 - std::ostreambuf_iterator< _CharT, _Traits >, [2702](#)
 - std::pointer_traits< _Ptr >, [2773](#)
 - std::pointer_traits< _Tp * >, [2774](#), [2775](#)
 - std::raw_storage_iterator< _OutputIterator, _Tp >, [2815](#)
 - std::set< _Key, _Compare, _Alloc >, [2892](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3126](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3155](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- Pointer Abstractions, [335](#)
 - allocate_shared, [338](#)
 - atomic_compare_exchange_strong, [339](#)
 - atomic_compare_exchange_strong_explicit, [340](#)
 - atomic_compare_exchange_weak, [341](#)
 - atomic_compare_exchange_weak_explicit, [342](#)
 - atomic_exchange, [343](#)
 - atomic_exchange_explicit, [343](#), [344](#)
 - atomic_is_lock_free, [344](#), [345](#)
 - atomic_load, [345](#)
 - atomic_load_explicit, [346](#)
 - atomic_store, [346](#), [347](#)
 - atomic_store_explicit, [347](#), [348](#)
 - const_pointer_cast, [348](#)
 - dynamic_pointer_cast, [348](#)
 - get_deleter, [349](#)
 - make_shared, [349](#)
 - operator<, [349](#), [350](#)
 - operator<=, [350](#)
 - operator>, [351](#)
 - operator>=, [352](#)
 - operator==, [350](#), [351](#)
 - reinterpret_pointer_cast, [352](#)
 - static_pointer_cast, [352](#)
 - swap, [353](#)
 - to_address, [353](#)
- pointer.h, [3596](#)
- pointer_to
 - std::pointer_traits< _Ptr >, [2773](#)
 - std::pointer_traits< _Tp * >, [2775](#), [2776](#)
- Poisson Distributions, [242](#)
 - operator<=, [243](#), [244](#)
 - operator>>, [244](#), [245](#)

- polar
 - Complex Numbers, [190](#)
 - std::tr1, [739](#)
- Policy-Based Data Structures, [144](#)
- policy_access_fnimps.hpp, [3563](#), [3564](#)
- Polymorphic memory resources, [354](#)
 - new_delete_resource, [355](#)
- pool_allocator.h, [3597](#)
- pop
 - std::priority_queue< _Tp, _Sequence, _Compare >, [2787](#)
 - std::queue< _Tp, _Sequence >, [2796](#)
 - std::stack< _Tp, _Sequence >, [2943](#)
- pop_back
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [875](#)
 - __gnu_parallel::RestrictedBoundedConcurrentQueue< _Tp >, [985](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1669](#)
 - std::deque< _Tp, _Alloc >, [2083](#)
 - std::list< _Tp, _Alloc >, [2380](#)
 - std::vector< _Tp, _Alloc >, [3250](#)
- pop_front
 - __gnu_parallel::RestrictedBoundedConcurrentQueue< _Tp >, [985](#)
 - std::deque< _Tp, _Alloc >, [2083](#)
 - std::forward_list< _Tp, _Alloc >, [2191](#)
 - std::list< _Tp, _Alloc >, [2381](#)
- pop_heap
 - Heap, [94](#), [95](#)
- pos_format
 - std::moneypunct< _CharT, _Intl >, [2496](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2503](#)
- pos_type
 - std::basic_ios< _CharT, _Traits >, [1242](#)
 - std::basic_streambuf< _CharT, _Traits >, [1498](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3261](#)
- position
 - std::match_results< _Bi_iter, _Alloc >, [2452](#)
- positive_sign
 - std::moneypunct< _CharT, _Intl >, [2496](#)
 - std::moneypunct_byname< _CharT, _Intl >, [2504](#)
- postypes.h, [3364](#)
- pow
 - Complex Numbers, [190](#)
 - math.h, [3636](#)
- power
 - SGL, [160](#)
- pptr
 - __gnu_cxx::enc_filebuf< _CharT >, [2128](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2955](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2973](#)
 - std::basic_filebuf< _CharT, _Traits >, [1130](#)
 - std::basic_streambuf< _CharT, _Traits >, [1502](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1709](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3266](#)
- precision
 - std::basic_fstream< _CharT, _Traits >, [1175](#)
 - std::basic_ifstream< _CharT, _Traits >, [1220](#), [1221](#)
 - std::basic_ios< _CharT, _Traits >, [1248](#), [1249](#)
 - std::basic_iostream< _CharT, _Traits >, [1290](#)
 - std::basic_istream< _CharT, _Traits >, [1331](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1371](#), [1372](#)
 - std::basic_ofstream< _CharT, _Traits >, [1407](#)
 - std::basic_ostream< _CharT, _Traits >, [1438](#), [1439](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1471](#), [1472](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1752](#)
 - std::ios_base, [2301](#), [2302](#)
- predefined_ops.h, [3364](#)
- prefix
 - std::match_results< _Bi_iter, _Alloc >, [2452](#)
- prefix_range
 - __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, [3073](#), [3074](#)
- prefix_search_node_update_imp.hpp, [3589](#)
- prev_permutation
 - Sorting, [82](#)
- print, [3685](#)
- print.h, [3365](#)
- priority_queue
 - __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, [2782](#)
 - std::priority_queue< _Tp, _Sequence, _Compare >, [2785](#), [2786](#)
- priority_queue.hpp, [3593](#)
- priority_queue_base_dispatch.hpp, [3583](#)
- probabilities
 - std::discrete_distribution< _IntType >, [2097](#)
- probe_fn_base.hpp, [3575](#)
- projected
 - std, [591](#)
- propagate_const, [3541](#)
- propagate_on_container_copy_assignment
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [758](#)
 - std::allocator_traits< _Alloc >, [1064](#)
 - std::allocator_traits< allocator< _Tp > >, [1069](#)
 - std::allocator_traits< allocator< void > >, [1075](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1078](#)
- propagate_on_container_move_assignment
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [758](#)
 - std::allocator_traits< _Alloc >, [1064](#)
 - std::allocator_traits< allocator< _Tp > >, [1070](#)

- std::allocator_traits< allocator< void > >, 1075
- std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1078
- propagate_on_container_swap
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 759
 - std::allocator_traits< _Alloc >, 1064
 - std::allocator_traits< allocator< _Tp > >, 1070
 - std::allocator_traits< allocator< void > >, 1075
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1079
- ptr_fun
 - Adaptors for pointers to functions, 318
- ptr_traits.h, 3365
- pubimbue
 - __gnu_cxx::enc_filebuf< _CharT >, 2128
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2955
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2973
 - std::basic_filebuf< _CharT, _Traits >, 1130
 - std::basic_streambuf< _CharT, _Traits >, 1503
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1710
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3266
- pubseekoff
 - __gnu_cxx::enc_filebuf< _CharT >, 2128
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
 - std::basic_filebuf< _CharT, _Traits >, 1131
 - std::basic_streambuf< _CharT, _Traits >, 1503
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1710
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3266
- pubseekpos
 - __gnu_cxx::enc_filebuf< _CharT >, 2129
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
 - std::basic_filebuf< _CharT, _Traits >, 1131
 - std::basic_streambuf< _CharT, _Traits >, 1503
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1710
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
- pubsetbuf
 - __gnu_cxx::enc_filebuf< _CharT >, 2129
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
 - std::basic_filebuf< _CharT, _Traits >, 1131
 - std::basic_streambuf< _CharT, _Traits >, 1504
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
- pubsync
 - __gnu_cxx::enc_filebuf< _CharT >, 2129
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
- std::basic_filebuf< _CharT, _Traits >, 1131
- std::basic_streambuf< _CharT, _Traits >, 1504
- std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
- std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
- push
 - std::priority_queue< _Tp, _Sequence, _Compare >, 2787
 - std::queue< _Tp, _Sequence >, 2796
 - std::stack< _Tp, _Sequence >, 2943
- push_back
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 875
 - std::basic_string< _CharT, _Traits, _Alloc >, 1669, 1670
 - std::deque< _Tp, _Alloc >, 2083
 - std::list< _Tp, _Alloc >, 2381
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2115
 - std::vector< _Tp, _Alloc >, 3250
- push_front
 - __gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >, 986
 - std::deque< _Tp, _Alloc >, 2083
 - std::forward_list< _Tp, _Alloc >, 2191
 - std::list< _Tp, _Alloc >, 2381
- push_heap
 - Heap, 95
- put
 - std::basic_fstream< _CharT, _Traits >, 1175
 - std::basic_istream< _CharT, _Traits >, 1290
 - std::basic_ofstream< _CharT, _Traits >, 1407
 - std::basic_ostream< _CharT, _Traits >, 1439
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1752
 - std::money_put< _CharT, _Outlter >, 2486, 2487
 - std::num_put< _CharT, _Outlter >, 2600–2605
 - std::time_put< _CharT, _Outlter >, 3043, 3044
 - std::time_put_byname< _CharT, _Outlter >, 3047
- put_money
 - std, 644
- put_time
 - std, 644
- putback
 - std::basic_fstream< _CharT, _Traits >, 1176
 - std::basic_ifstream< _CharT, _Traits >, 1221
 - std::basic_istream< _CharT, _Traits >, 1290
 - std::basic_istream< _CharT, _Traits >, 1331
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1372
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1753
- pword
 - std::basic_fstream< _CharT, _Traits >, 1176

- std::basic_ifstream< _CharT, _Traits >, 1221
- std::basic_ios< _CharT, _Traits >, 1249
- std::basic_iostream< _CharT, _Traits >, 1291
- std::basic_istream< _CharT, _Traits >, 1332
- std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1372
- std::basic_ofstream< _CharT, _Traits >, 1408
- std::basic_ostream< _CharT, _Traits >, 1439
- std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1472
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1753
- std::ios_base, 2302
- qsb_steals
 - __gnu_parallel::Settings, 1042
- quadratic_probe_fn_imp.hpp, 3575
- queue, 3685
 - std::queue< _Tp, _Sequence >, 2795
- queue.h, 3678
 - _GLIBCXX_VOLATILE, 3679
- quicksort.h, 3679
- quiet_NaN
 - std::numeric_limits< _Tp >, 2608
- quoted_string.h, 3366
- r_erase_fn_imps.hpp, 3550, 3551
- radix
 - std::__numeric_limits_base, 817
 - std::numeric_limits< _Tp >, 2610
 - std::numeric_limits< bool >, 2614
 - std::numeric_limits< char >, 2618
 - std::numeric_limits< char16_t >, 2622
 - std::numeric_limits< char32_t >, 2626
 - std::numeric_limits< double >, 2630
 - std::numeric_limits< float >, 2634
 - std::numeric_limits< int >, 2638
 - std::numeric_limits< long >, 2642
 - std::numeric_limits< long double >, 2646
 - std::numeric_limits< long long >, 2650
 - std::numeric_limits< short >, 2654
 - std::numeric_limits< signed char >, 2658
 - std::numeric_limits< unsigned char >, 2662
 - std::numeric_limits< unsigned int >, 2666
 - std::numeric_limits< unsigned long >, 2670
 - std::numeric_limits< unsigned long long >, 2674
 - std::numeric_limits< unsigned short >, 2678
 - std::numeric_limits< wchar_t >, 2682
- random, 3685
- Random Number Distributions, 238
- Random Number Generation, 237
 - generate_canonical, 238
- Random Number Generators, 248
 - minstd_rand, 249
 - minstd_rand0, 249
 - mt19937, 249
 - mt19937_64, 250
 - operator<<, 250
 - philox4x32, 250
 - philox4x64, 250
- Random Number Utilities, 251
- random.h, 3366
- random.tcc, 3369, 3373
- random_number.h, 3679
- random_sample
 - SGI, 161
- random_sample_n
 - SGI, 161
- random_shuffle
 - Mutating, 31
- random_shuffle.h, 3679
- random_shuffle_minimal_n
 - __gnu_parallel::Settings, 1042
- range_access.h, 3375
- ranged_hash_fn.hpp, 3576
- ranged_probe_fn.hpp, 3576
- Ranges, 256
- ranges, 3686
- ranges_algo.h, 3376
- ranges_algobase.h, 3381
- ranges_base.h, 3382
 - enable_view, 3384
- ranges_cmp.h, 3385
- ranges_uninitialized.h, 3385
- ranges_util.h, 3385
- ratio, 3689, 3691
- ratio_add
 - Rational Arithmetic, 366
- ratio_divide
 - Rational Arithmetic, 366
- ratio_multiply
 - Rational Arithmetic, 366
- ratio_subtract
 - Rational Arithmetic, 366
- Rational Arithmetic, 364
 - ratio_add, 366
 - ratio_divide, 366
 - ratio_multiply, 366
 - ratio_subtract, 366
- rb_tree, 3598
- rb_tree.hpp, 3584
- rbegin
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 875
 - std, 645, 646
 - std::basic_string< _CharT, _Traits, _Alloc >, 1670
 - std::deque< _Tp, _Alloc >, 2084
 - std::list< _Tp, _Alloc >, 2381, 2382
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2439

- std::multimap< _Key, _Tp, _Compare, _Alloc >, [2532](#)
- std::multiset< _Key, _Compare, _Alloc >, [2553](#)
- std::set< _Key, _Compare, _Alloc >, [2905](#)
- std::vector< _Tp, _Alloc >, [3250](#)
- rc.hpp, [3584](#)
- rc_binomial_heap.hpp, [3584](#)
- rc_string_base.h, [3598](#)
- rdbuf
 - std::basic_fstream< _CharT, _Traits >, [1177](#)
 - std::basic_ifstream< _CharT, _Traits >, [1222](#)
 - std::basic_ios< _CharT, _Traits >, [1249](#), [1250](#)
 - std::basic_iostream< _CharT, _Traits >, [1291](#), [1292](#)
 - std::basic_istream< _CharT, _Traits >, [1332](#), [1333](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1373](#)
 - std::basic_ofstream< _CharT, _Traits >, [1408](#), [1409](#)
 - std::basic_ostream< _CharT, _Traits >, [1440](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1473](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1754](#)
- rdstate
 - std::basic_fstream< _CharT, _Traits >, [1177](#)
 - std::basic_ifstream< _CharT, _Traits >, [1222](#)
 - std::basic_ios< _CharT, _Traits >, [1250](#)
 - std::basic_iostream< _CharT, _Traits >, [1292](#)
 - std::basic_istream< _CharT, _Traits >, [1333](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1373](#)
 - std::basic_ofstream< _CharT, _Traits >, [1409](#)
 - std::basic_ostream< _CharT, _Traits >, [1441](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1473](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1754](#)
- read
 - std::basic_fstream< _CharT, _Traits >, [1178](#)
 - std::basic_ifstream< _CharT, _Traits >, [1223](#)
 - std::basic_iostream< _CharT, _Traits >, [1292](#)
 - std::basic_istream< _CharT, _Traits >, [1333](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1374](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1755](#)
- readsome
 - std::basic_fstream< _CharT, _Traits >, [1178](#)
 - std::basic_ifstream< _CharT, _Traits >, [1223](#)
 - std::basic_iostream< _CharT, _Traits >, [1293](#)
 - std::basic_istream< _CharT, _Traits >, [1334](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1374](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1755](#)
- ready
 - std::match_results< _Bi_iter, _Alloc >, [2452](#)
- rebind
 - std::pointer_traits< _Ptr >, [2773](#)
 - std::pointer_traits< _Tp * >, [2775](#)
- reduce
 - Generalized Numeric operations, [15](#), [16](#)
- ref
 - std::reference_wrapper< _Tp >, [2833](#)
- reference
 - __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_iterator, Iterator, _Alloc >, [1775](#)
 - __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_iterator, Iterator, _Alloc >, [1780](#)
 - __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, [1790](#)
 - __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, [1793](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, [2345](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, [2349](#)
 - std::back_insert_iterator< _Container >, [1107](#)
 - std::front_insert_iterator< _Container >, [2201](#)
 - std::insert_iterator< _Container >, [2291](#)
 - std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, [2331](#)
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
 - std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, [2340](#)
 - std::ostream_iterator< _Tp, _CharT, _Traits >, [2699](#)
 - std::ostreambuf_iterator< _CharT, _Traits >, [2702](#)
 - std::raw_storage_iterator< _OutputIterator, _Tp >, [2815](#)
 - std::set< _Key, _Compare, _Alloc >, [2892](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3126](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3155](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- refwrap.h, [3386](#)
- regex, [3691](#)
 - Regular Expressions, [260](#)
- regex.h, [3387](#)
- regex.tcc, [3389](#)
- regex_automaton.h, [3389](#)
- regex_automaton.tcc, [3390](#)
- regex_compiler.h, [3390](#)
- regex_compiler.tcc, [3391](#)
- regex_constants.h, [3391](#)
- regex_error

- std::regex_error, [2834](#)
- regex_error.h, [3392](#)
- regex_executor.h, [3393](#)
- regex_executor.tcc, [3393](#)
- regex_iterator
 - std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, [2836](#)
- regex_match
 - Regular Expressions, [264–268](#)
- regex_replace
 - Regular Expressions, [269–272](#)
- regex_scanner.h, [3394](#)
- regex_scanner.tcc, [3394](#)
- regex_search
 - Regular Expressions, [273–276](#)
- regex_token_iterator
 - std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, [2839–2841](#)
- regex_traits
 - std::regex_traits< _Ch_type >, [2843](#)
- register_callback
 - std::basic_fstream< _CharT, _Traits >, [1179](#)
 - std::basic_ifstream< _CharT, _Traits >, [1224](#)
 - std::basic_ios< _CharT, _Traits >, [1250](#)
 - std::basic_iostream< _CharT, _Traits >, [1294](#)
 - std::basic_istream< _CharT, _Traits >, [1335](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1375](#)
 - std::basic_ofstream< _CharT, _Traits >, [1409](#)
 - std::basic_ostream< _CharT, _Traits >, [1441](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1474](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1756](#)
 - std::ios_base, [2302](#)
- Regular Expressions, [257](#)
 - cregex_token_iterator, [260](#)
 - csub_match, [260](#)
 - operator<<, [261](#)
 - operator<=>, [261](#), [262](#)
 - operator==, [262–264](#)
 - regex, [260](#)
 - regex_match, [264–268](#)
 - regex_replace, [269–272](#)
 - regex_search, [273–276](#)
 - sregex_token_iterator, [260](#)
 - ssub_match, [260](#)
 - swap, [277](#)
 - wcregex_token_iterator, [260](#)
 - wcsub_match, [260](#)
 - wregex, [260](#)
 - wsregex_token_iterator, [260](#)
 - wssub_match, [261](#)
- rehash
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3146](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3171](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3195](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3220](#)
- reinterpret_pointer_cast
 - Pointer Abstractions, [352](#)
- release
 - std::auto_ptr< _Tp >, [1104](#)
 - std::unique_ptr< _Tp, _Dp >, [3111](#)
 - std::unique_ptr< _Tp[], _Dp >, [3116](#)
- remove
 - Mutating, [32](#)
 - std::forward_list< _Tp, _Alloc >, [2192](#)
 - std::list< _Tp, _Alloc >, [2382](#)
- remove_all_extents_t
 - Metaprogramming, [361](#)
- remove_copy
 - Mutating, [32](#)
- remove_copy_if
 - Mutating, [33](#)
- remove_extent_t
 - Metaprogramming, [361](#)
- remove_if
 - Mutating, [33](#)
 - std::forward_list< _Tp, _Alloc >, [2192](#)
 - std::list< _Tp, _Alloc >, [2382](#)
- remove_pointer_t
 - Metaprogramming, [361](#)
- remove_reference_t
 - Metaprogramming, [361](#)
- rend
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [876](#)
 - std, [646](#), [647](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1670](#), [1671](#)
 - std::deque< _Tp, _Alloc >, [2084](#)
 - std::list< _Tp, _Alloc >, [2382](#)
 - std::map< _Key, _Tp, _Compare, _Alloc >, [2440](#)
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, [2532](#)
 - std::multiset< _Key, _Compare, _Alloc >, [2554](#)
 - std::set< _Key, _Compare, _Alloc >, [2906](#)
 - std::vector< _Tp, _Alloc >, [3251](#)
- replace
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [876–882](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1569–1579](#)
 - Mutating, [34](#)

- std::basic_string< _CharT, _Traits, _Alloc >, [1671–1684](#)
- replace_copy
 - std, [647](#)
- replace_copy_if
 - Mutating, [34](#)
- replace_if
 - Mutating, [35](#)
- replace_minimal_n
 - __gnu_parallel:: Settings, [1042](#)
- requires_hosted.h, [3394](#)
- reserve
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [883](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1580](#), [1581](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1685](#), [1686](#)
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3146](#)
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3172](#)
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3196](#)
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3221](#)
 - std::vector< _Tp, _Alloc >, [3251](#)
- reset
 - std::auto_ptr< _Tp >, [1104](#)
 - std::bernoulli_distribution, [1771](#)
 - std::binomial_distribution< _IntType >, [1801](#)
 - std::bitset< _Nb >, [1819](#)
 - std::cauchy_distribution< _RealType >, [1826](#)
 - std::chi_squared_distribution< _RealType >, [1851](#)
 - std::discrete_distribution< _IntType >, [2098](#)
 - std::exponential_distribution< _RealType >, [2159](#)
 - std::extreme_value_distribution< _RealType >, [2162](#)
 - std::fisher_f_distribution< _RealType >, [2172](#)
 - std::gamma_distribution< _RealType >, [2219](#)
 - std::geometric_distribution< _IntType >, [2222](#)
 - std::lognormal_distribution< _RealType >, [2408](#)
 - std::negative_binomial_distribution< _IntType >, [2563](#)
 - std::normal_distribution< _RealType >, [2570](#)
 - std::piecewise_constant_distribution< _RealType >, [2763](#)
 - std::piecewise_linear_distribution< _RealType >, [2766](#)
 - std::poisson_distribution< _IntType >, [2779](#)
 - std::student_t_distribution< _RealType >, [2989](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2115](#)
 - std::uniform_int_distribution< _IntType >, [3102](#)
 - std::uniform_real_distribution< _RealType >, [3105](#)
 - std::unique_ptr< _Tp, _Dp >, [3111](#)
 - std::unique_ptr< _Tp[], _Dp >, [3117](#)
 - std::weibull_distribution< _RealType >, [3279](#)
- resetiosflags
 - std, [648](#)
- resize
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [883](#), [884](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1582](#)
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, [2275](#)
 - Numeric Arrays, [235](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1686](#), [1687](#)
 - std::deque< _Tp, _Alloc >, [2084](#)
 - std::forward_list< _Tp, _Alloc >, [2192](#), [2193](#)
 - std::list< _Tp, _Alloc >, [2383](#)
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, [2115](#)
 - std::vector< _Tp, _Alloc >, [3251](#), [3252](#)
- resize_fn_imps.hpp, [3571](#)
- resize_no_store_hash_fn_imps.hpp, [3571](#)
- resize_policy.hpp, [3564](#)
- resize_store_hash_fn_imps.hpp, [3571](#)
- result_of_t
 - Metaprogramming, [361](#)
- result_type
 - __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, [1783](#)
 - __gnu_cxx::equal_to< _Tp >, [2144](#)
 - __gnu_cxx::project1st< _Arg1, _Arg2 >, [2789](#)
 - __gnu_cxx::project2nd< _Arg1, _Arg2 >, [2789](#)
 - __gnu_cxx::select1st< _Pair >, [2881](#)
 - __gnu_cxx::select2nd< _Pair >, [2881](#)
 - __gnu_cxx::subtractive_rng, [3000](#)
 - __gnu_cxx::unary_compose< _Operation1, _Operation2 >, [3095](#)
 - __gnu_parallel:: EqualFromLess< _T1, _T2, _Compare >, [905](#)
 - __gnu_parallel:: EqualTo< _T1, _T2 >, [906](#)
 - __gnu_parallel:: Less< _T1, _T2 >, [932](#)
 - __gnu_parallel:: Lexicographic< _T1, _T2, _Compare >, [934](#)
 - __gnu_parallel:: LexicographicReverse< _T1, _T2, _Compare >, [935](#)
 - __gnu_parallel:: Multiplies< _Tp1, _Tp2, _Result >, [961](#)
 - __gnu_parallel:: Plus< _Tp1, _Tp2, _Result >, [973](#)
 - __gnu_parallel:: binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, [766](#)
 - __gnu_parallel:: binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, [767](#)

- `__gnu_parallel::__unary_negate< _Predicate, argument_type >`, 834
- `std::bernoulli_distribution`, 1769
- `std::binary_function< _Arg1, _Arg2, _Result >`, 1786
- `std::binary_negate< _Predicate >`, 1796
- `std::binder1st< _Operation >`, 1797
- `std::binder2nd< _Operation >`, 1799
- `std::binomial_distribution< _IntType >`, 1800
- `std::cauchy_distribution< _RealType >`, 1825
- `std::chi_squared_distribution< _RealType >`, 1851
- `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 1923
- `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 1924
- `std::const_mem_fun_ref_t< _Ret, _Tp >`, 1925
- `std::const_mem_fun_t< _Ret, _Tp >`, 1926
- `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2092
- `std::discrete_distribution< _IntType >`, 2097
- `std::divides< _Tp >`, 2100
- `std::divides< void >`, 2101
- `std::equal_to< _Tp >`, 2145
- `std::equal_to< void >`, 2147
- `std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >`, 2715
- `std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >`, 2717
- `std::exponential_distribution< _RealType >`, 2158
- `std::extreme_value_distribution< _RealType >`, 2161
- `std::fisher_f_distribution< _RealType >`, 2172
- `std::gamma_distribution< _RealType >`, 2218
- `std::geometric_distribution< _IntType >`, 2221
- `std::greater< _Tp >`, 2235
- `std::greater< void >`, 2237
- `std::greater_equal< _Tp >`, 2238
- `std::greater_equal< void >`, 2239
- `std::hash< __gnu_cxx::throw_value_limit >`, 2247
- `std::hash< __gnu_cxx::throw_value_random >`, 2248
- `std::independent_bits_engine< _RandomNumberEngine, __w, _IntType >`, 2280
- `std::less< _Tp >`, 2352
- `std::less< void >`, 2354
- `std::less_equal< _Tp >`, 2355
- `std::less_equal< void >`, 2356
- `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, 2359
- `std::logical_and< _Tp >`, 2400
- `std::logical_and< void >`, 2401
- `std::logical_not< _Tp >`, 2402
- `std::logical_not< void >`, 2403
- `std::logical_or< _Tp >`, 2404
- `std::logical_or< void >`, 2405
- `std::lognormal_distribution< _RealType >`, 2407
- `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 2457
- `std::mem_fun1_t< _Ret, _Tp, _Arg >`, 2458
- `std::mem_fun_ref_t< _Ret, _Tp >`, 2459
- `std::mem_fun_t< _Ret, _Tp >`, 2460
- `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`, 2464
- `std::minus< _Tp >`, 2473
- `std::minus< void >`, 2474
- `std::modulus< _Tp >`, 2476
- `std::modulus< void >`, 2477
- `std::multiplies< _Tp >`, 2535
- `std::multiplies< void >`, 2536
- `std::negate< _Tp >`, 2560
- `std::negate< void >`, 2561
- `std::negative_binomial_distribution< _IntType >`, 2562
- `std::normal_distribution< _RealType >`, 2569
- `std::not_equal_to< _Tp >`, 2572
- `std::not_equal_to< void >`, 2574
- `std::owner_less< shared_ptr< _Tp > >`, 2716
- `std::owner_less< void >`, 2717
- `std::owner_less< weak_ptr< _Tp > >`, 2718
- `std::piecewise_constant_distribution< _RealType >`, 2762
- `std::piecewise_linear_distribution< _RealType >`, 2765
- `std::plus< _Tp >`, 2768
- `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`, 2771
- `std::pointer_to_unary_function< _Arg, _Result >`, 2772
- `std::poisson_distribution< _IntType >`, 2777
- `std::random_device`, 2800
- `std::seed_seq`, 2880
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2925
- `std::student_t_distribution< _RealType >`, 2988
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 2996
- `std::unary_function< _Arg, _Result >`, 3096
- `std::unary_negate< _Predicate >`, 3097
- `std::uniform_int_distribution< _IntType >`, 3101
- `std::uniform_real_distribution< _RealType >`, 3103
- `std::weibull_distribution< _RealType >`, 3278
- `rethrow_exception`
- `exception_ptr.h`, 3330
- Exceptions, 136
- `rethrow_if_nested`
- Exceptions, 136
- `rethrow_nested`
- `std::nested_exception`, 2565
- `return_temporary_buffer`
- `std`, 648
- `reverse`

- Mutating, [35](#)
 - std::forward_list< _Tp, _Alloc >, [2193](#)
 - std::list< _Tp, _Alloc >, [2383](#)
- reverse_copy
 - Mutating, [36](#)
- reverse_iteration
 - __gnu_pbds::container_traits< Cntnr >, [1941](#)
- reverse_iterator
 - std::reverse_iterator< _Iterator >, [2854](#)
 - std::set< _Key, _Compare, _Alloc >, [2892](#)
- rfind
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, [884–886](#)
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, [1582–1586](#)
 - std::basic_string< _CharT, _Traits, _Alloc >, [1687–1690](#)
- riemann_zeta
 - Mathematical Special Functions, [217](#)
 - TR1 Mathematical Special Functions, [255](#)
- riemann_zetaf
 - Mathematical Special Functions, [218](#)
- riemann_zetal
 - Mathematical Special Functions, [218](#)
- right
 - std, [648](#)
 - std::basic_fstream< _CharT, _Traits >, [1190](#)
 - std::basic_ifstream< _CharT, _Traits >, [1233](#)
 - std::basic_ios< _CharT, _Traits >, [1258](#)
 - std::basic_iostream< _CharT, _Traits >, [1305](#)
 - std::basic_istream< _CharT, _Traits >, [1344](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1385](#)
 - std::basic_ofstream< _CharT, _Traits >, [1418](#)
 - std::basic_ostream< _CharT, _Traits >, [1450](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1483](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1767](#)
 - std::ios_base, [2308](#)
- rope, [3599](#)
- ropeimpl.h, [3602](#)
- rotate
 - Mutating, [36](#)
- rotate_copy
 - Mutating, [37](#)
- rotate_fn_imps.hpp, [3551](#)
- round
 - Time, [385](#), [386](#)
- round_error
 - std::numeric_limits< _Tp >, [2608](#)
- round_indeterminate
 - std, [592](#)
- round_style
 - std::__numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2611](#)
 - std::numeric_limits< bool >, [2614](#)
 - std::numeric_limits< char >, [2618](#)
 - std::numeric_limits< char16_t >, [2622](#)
 - std::numeric_limits< char32_t >, [2626](#)
 - std::numeric_limits< double >, [2630](#)
 - std::numeric_limits< float >, [2634](#)
 - std::numeric_limits< int >, [2638](#)
 - std::numeric_limits< long >, [2642](#)
 - std::numeric_limits< long double >, [2646](#)
 - std::numeric_limits< long long >, [2650](#)
 - std::numeric_limits< short >, [2654](#)
 - std::numeric_limits< signed char >, [2658](#)
 - std::numeric_limits< unsigned char >, [2662](#)
 - std::numeric_limits< unsigned int >, [2666](#)
 - std::numeric_limits< unsigned long >, [2670](#)
 - std::numeric_limits< unsigned long long >, [2674](#)
 - std::numeric_limits< unsigned short >, [2678](#)
 - std::numeric_limits< wchar_t >, [2682](#)
- round_to_nearest
 - std, [592](#)
- round_toward_infinity
 - std, [592](#)
- round_toward_neg_infinity
 - std, [592](#)
- round_toward_zero
 - std, [592](#)
- runtime_error
 - std::runtime_error, [2863](#)
- safe_base.h, [3524](#)
- safe_container.h, [3524](#)
- safe_iterator.h, [3525](#)
- safe_iterator.tcc, [3526](#)
- safe_local_iterator.h, [3527](#)
- safe_local_iterator.tcc, [3527](#)
- safe_sequence.h, [3528](#)
- safe_sequence.tcc, [3528](#)
- safe_unordered_base.h, [3528](#)
- safe_unordered_container.h, [3528](#)
- safe_unordered_container.tcc, [3529](#)
- sample
 - std, [648](#)
 - std::experimental, [715](#)
- sample_probe_fn
 - __gnu_pbds::sample_probe_fn, [2864](#)
- sample_probe_fn.hpp, [3576](#)
- sample_range_hashing
 - __gnu_pbds::sample_range_hashing, [2865](#)
 - __gnu_pbds::sample_resize_policy, [2869](#)
 - __gnu_pbds::sample_resize_trigger, [2872](#)
 - __gnu_pbds::sample_size_policy, [2873](#)
- sample_range_hashing.hpp, [3577](#)

- sample_ranged_hash_fn
 - __gnu_pbds::sample_ranged_hash_fn, 2866
- sample_ranged_hash_fn.hpp, 3577
- sample_ranged_probe_fn.hpp, 3577
- sample_resize_policy
 - __gnu_pbds::sample_resize_policy, 2868
- sample_resize_policy.hpp, 3585
- sample_resize_trigger
 - __gnu_pbds::sample_resize_trigger, 2870
- sample_resize_trigger.hpp, 3586
- sample_size_policy
 - __gnu_pbds::sample_size_policy, 2873
- sample_size_policy.hpp, 3586
- sample_tree_node_update.hpp, 3589
- sample_trie_access_traits.hpp, 3589
- sample_trie_node_update
 - __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >, 2875
- sample_trie_node_update.hpp, 3589
- sample_update_policy
 - __gnu_pbds::sample_update_policy, 2876
- sample_update_policy.hpp, 3580
- Sampling Distributions, 245
- sat_arith.h, 3394
- saturating_add
 - std, 649
- saturating_cast
 - std, 649
- saturating_div
 - std, 649
- saturating_mul
 - std, 649
- saturating_sub
 - std, 649
- sbumpc
 - __gnu_cxx::enc_filebuf< _CharT >, 2129
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2975
 - std::basic_filebuf< _CharT, _Traits >, 1132
 - std::basic_streambuf< _CharT, _Traits >, 1504
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
- scan_is
 - std::__ctype_abstract_base< _CharT >, 784
 - std::ctype< _CharT >, 1961
 - std::ctype< char >, 1986
 - std::ctype< wchar_t >, 2006
 - std::ctype_byname< _CharT >, 2020
 - std::ctype_byname< char >, 2041
- scan_not
 - std::__ctype_abstract_base< _CharT >, 785
 - std::ctype< _CharT >, 1962
 - std::ctype< char >, 1987, 1988
- std::ctype< wchar_t >, 2006
- std::ctype_byname< _CharT >, 2021
- std::ctype_byname< char >, 2042
- scientific
 - std, 649
 - std::basic_fstream< _CharT, _Traits >, 1190
 - std::basic_ifstream< _CharT, _Traits >, 1233
 - std::basic_ios< _CharT, _Traits >, 1258
 - std::basic_iostream< _CharT, _Traits >, 1305
 - std::basic_istream< _CharT, _Traits >, 1344
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1385
 - std::basic_ofstream< _CharT, _Traits >, 1418
 - std::basic_ostream< _CharT, _Traits >, 1450
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1483
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1767
 - std::ios_base, 2308
- scoped_allocator, 3692
- search
 - Non-Mutating, 59, 60
 - std, 649
 - std::__parallel, 686
- search.h, 3680
- search_minimal_n
 - __gnu_parallel::Settings, 1043
- search_n
 - Non-Mutating, 61
- second
 - __gnu_cxx::pair< _T1, _T2 >, 2724
 - __gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 928
 - std::pair< _T1, _T2 >, 2730
- second_argument_type
 - __gnu_cxx::equal_to< _Tp >, 2144
 - __gnu_cxx::project1st< _Arg1, _Arg2 >, 2789
 - __gnu_cxx::project2nd< _Arg1, _Arg2 >, 2790
 - __gnu_parallel::EqualFromLess< _T1, _T2, _Compare >, 905
 - __gnu_parallel::EqualTo< _T1, _T2 >, 906
 - __gnu_parallel::Less< _T1, _T2 >, 932
 - __gnu_parallel::Lexicographic< _T1, _T2, _Compare >, 934
 - __gnu_parallel::LexicographicReverse< _T1, _T2, _Compare >, 935
 - __gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >, 961
 - __gnu_parallel::Plus< _Tp1, _Tp2, _Result >, 973
- std::binary_function< _Arg1, _Arg2, _Result >, 1786
- std::binary_negate< _Predicate >, 1796
- std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1923
- std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1924

- std::divides< _Tp >, 2100
- std::divides< void >, 2101
- std::equal_to< _Tp >, 2145
- std::equal_to< void >, 2147
- std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2715
- std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2718
- std::greater< _Tp >, 2235
- std::greater< void >, 2237
- std::greater_equal< _Tp >, 2238
- std::greater_equal< void >, 2239
- std::less< _Tp >, 2352
- std::less< void >, 2354
- std::less_equal< _Tp >, 2355
- std::less_equal< void >, 2356
- std::logical_and< _Tp >, 2400
- std::logical_and< void >, 2401
- std::logical_or< _Tp >, 2404
- std::logical_or< void >, 2405
- std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2457
- std::mem_fun1_t< _Ret, _Tp, _Arg >, 2458
- std::minus< _Tp >, 2473
- std::minus< void >, 2474
- std::modulus< _Tp >, 2476
- std::modulus< void >, 2477
- std::multiplies< _Tp >, 2535
- std::multiplies< void >, 2536
- std::not_equal_to< _Tp >, 2572
- std::not_equal_to< void >, 2574
- std::owner_less< shared_ptr< _Tp > >, 2716
- std::owner_less< void >, 2717
- std::owner_less< weak_ptr< _Tp > >, 2718
- std::plus< _Tp >, 2769
- std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2771
- second_type
 - __gnu_cxx::pair< _T1, _T2 >, 2721
 - __gnu_parallel::iterator_pair< _Iterator1, _Iterator2, _IteratorCategory >, 925
 - std::pair< _T1, _T2 >, 2727
- seconds
 - Time, 376
- seed
 - std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2094
 - std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2282
 - std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2360
 - std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2759
 - std::shuffle_order_engine< _RandomNumberEngine, __k >, 2927
 - std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2997
- seed_seq
 - std::seed_seq, 2880
- seekdir
 - std::basic_fstream< _CharT, _Traits >, 1149
 - std::basic_ifstream< _CharT, _Traits >, 1200
 - std::basic_ios< _CharT, _Traits >, 1242
 - std::basic_iostream< _CharT, _Traits >, 1266
 - std::basic_istream< _CharT, _Traits >, 1313
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1353
 - std::basic_ofstream< _CharT, _Traits >, 1392
 - std::basic_ostream< _CharT, _Traits >, 1426
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1458
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1728
 - std::ios_base, 2299
- seekg
 - std::basic_fstream< _CharT, _Traits >, 1179, 1180
 - std::basic_ifstream< _CharT, _Traits >, 1224, 1225
 - std::basic_iostream< _CharT, _Traits >, 1294
 - std::basic_istream< _CharT, _Traits >, 1335
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1375, 1376
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756, 1757
- seekoff
 - __gnu_cxx::enc_filebuf< _CharT >, 2129
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2975
 - std::basic_filebuf< _CharT, _Traits >, 1132
 - std::basic_streambuf< _CharT, _Traits >, 1504
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3268
- seekp
 - std::basic_fstream< _CharT, _Traits >, 1180, 1181
 - std::basic_iostream< _CharT, _Traits >, 1295
 - std::basic_ofstream< _CharT, _Traits >, 1410
 - std::basic_ostream< _CharT, _Traits >, 1441, 1442
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, 1474
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
- seekpos
 - __gnu_cxx::enc_filebuf< _CharT >, 2130
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2975
 - std::basic_filebuf< _CharT, _Traits >, 1132
 - std::basic_streambuf< _CharT, _Traits >, 1505
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1712

- std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3268
- select_on_container_copy_construction
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, 763
 - std::allocator_traits< _Alloc >, 1067
 - std::allocator_traits< allocator< _Tp > >, 1073
 - std::allocator_traits< allocator< void > >, 1076
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1082
- semaphore, 3692
- semaphore_base.h, 3395
- sentry
 - std::basic_istream< _CharT, _Traits >::sentry, 2883
 - std::basic_ostream< _CharT, _Traits >::sentry, 2884
- Sequences, 127
- sequential
 - __gnu_parallel, 432
- set, 3692, 3693
 - __gnu_parallel::_Settings, 1039
 - std::bitset< _Nb >, 1819
 - std::set< _Key, Compare, _Alloc >, 2893–2895
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2115
- Set Operations, 97
 - includes, 98
 - set_difference, 99, 100
 - set_intersection, 100, 101
 - set_symmetric_difference, 102
 - set_union, 103, 104
- set.h, 3529
- set_counter
 - std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2759
- set_default_resource
 - memory_resource, 3644
- set_difference
 - Set Operations, 99, 100
- set_difference_minimal_n
 - __gnu_parallel::_Settings, 1043
- set_intersection
 - Set Operations, 100, 101
- set_intersection_minimal_n
 - __gnu_parallel::_Settings, 1043
- set_load
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger<std::basic_istream< _CharT, _Traits, _Alloc >, External_Load_Access, Size_Type >, 1830
- set_loads
 - __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2264
 - __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2275
- set_new_handler
 - std, 650
- set_num_threads
 - __gnu_parallel::balanced_quicksort_tag, 1116
- __gnu_parallel::balanced_tag, 1117
- __gnu_parallel::default_parallel_tag, 2058
- __gnu_parallel::exact_tag, 2153
- __gnu_parallel::multiway_mergesort_exact_tag, 2556
- __gnu_parallel::multiway_mergesort_sampling_tag, 2557
- __gnu_parallel::multiway_mergesort_tag, 2558
- __gnu_parallel::omp_loop_static_tag, 2693
- __gnu_parallel::omp_loop_tag, 2694
- __gnu_parallel::parallel_tag, 2734
- __gnu_parallel::quicksort_tag, 2797
- __gnu_parallel::sampling_tag, 2878
- __gnu_parallel::unbalanced_tag, 3098
- set_operations.h, 3680
- set_symmetric_difference
 - Set Operations, 102
- set_symmetric_difference_minimal_n
 - __gnu_parallel::_Settings, 1043
- set_terminate
 - Exceptions, 136
- set_union
 - Set Operations, 103, 104
- set_union_minimal_n
 - __gnu_parallel::_Settings, 1043
- setbase
 - std, 650
- setbuf
 - __gnu_cxx::enc_filebuf< _CharT >, 2130
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2958
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2975
 - std::basic_filebuf< _CharT, _Traits >, 1132
 - std::basic_streambuf< _CharT, _Traits >, 1505
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1712
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3268
- setf
 - std::basic_fstream< _CharT, _Traits >, 1181
 - std::basic_ifstream< _CharT, _Traits >, 1225
 - std::basic_ios< _CharT, _Traits >, 1251
 - std::basic_iostream< _CharT, _Traits >, 1296
 - std::basic_istream< _CharT, _Traits >, 1336
 - std::basic_istream< _CharT, _Traits, _Alloc >, 1376
 - std::basic_ofstream< _CharT, _Traits >, 1410, 1411
 - std::basic_ostream< _CharT, _Traits >, 1442
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
 - std::ios_base, 2303
- setfill
 - std, 650
- setg

- `__gnu_cxx::enc_filebuf< _CharT >`, 2130
- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2958
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2976
- `std::basic_filebuf< _CharT, _Traits >`, 1133
- `std::basic_streambuf< _CharT, _Traits >`, 1505
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1712
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3269
- setiosflags
 - std, 650
- setp
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2131
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2959
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2976
 - `std::basic_filebuf< _CharT, _Traits >`, 1133
 - `std::basic_streambuf< _CharT, _Traits >`, 1506
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1713
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3269
- setprecision
 - std, 651
- setstate
 - `std::basic_fstream< _CharT, _Traits >`, 1182
 - `std::basic_ifstream< _CharT, _Traits >`, 1226
 - `std::basic_ios< _CharT, _Traits >`, 1251
 - `std::basic_iostream< _CharT, _Traits >`, 1296
 - `std::basic_istream< _CharT, _Traits >`, 1336
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1377
 - `std::basic_ofstream< _CharT, _Traits >`, 1411
 - `std::basic_ostream< _CharT, _Traits >`, 1443
 - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1475
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1758
- settings.h, 3681
 - `_GLIBCXX_PARALLEL_CONDITION`, 3682
- setw
 - std, 651
- sgetc
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2131
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2959
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2977
 - `std::basic_filebuf< _CharT, _Traits >`, 1134
 - `std::basic_streambuf< _CharT, _Traits >`, 1506
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1713
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3269
- sgetn
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2131
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2959
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2977
 - `std::basic_filebuf< _CharT, _Traits >`, 1134
- `std::basic_streambuf< _CharT, _Traits >`, 1506
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1713
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3270
- SGL, 153
 - `_Find_first`, 157
 - `_Find_next`, 157
 - `_Unchecked_flip`, 157
 - `_Unchecked_reset`, 158
 - `_Unchecked_set`, 158
 - `_Unchecked_test`, 158
 - `__median`, 156
 - `compose1`, 158
 - `compose2`, 158
 - `constant0`, 158
 - `constant1`, 159
 - `constant2`, 159
 - `copy_n`, 159
 - `distance`, 159
 - `identity_element`, 160
 - `lexicographical_compare_3way`, 160
 - `power`, 160
 - `random_sample`, 161
 - `random_sample_n`, 161
 - `uninitialized_copy_n`, 162
- shared_future
 - `std::shared_future< _Res >`, 2908
 - `std::shared_future< _Res & >`, 2910
 - `std::shared_future< void >`, 2912
- shared_mutex, 3693
- shared_ptr
 - `std::shared_ptr< _Tp >`, 2916–2922
- shared_ptr.h, 3395, 3396
- shared_ptr_atomic.h, 3398
- shared_ptr_base.h, 3400
- shift
 - Numeric Arrays, 236
- showbase
 - std, 651
 - `std::basic_fstream< _CharT, _Traits >`, 1190
 - `std::basic_ifstream< _CharT, _Traits >`, 1234
 - `std::basic_ios< _CharT, _Traits >`, 1258
 - `std::basic_iostream< _CharT, _Traits >`, 1305
 - `std::basic_istream< _CharT, _Traits >`, 1344
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1385
 - `std::basic_ofstream< _CharT, _Traits >`, 1418
 - `std::basic_ostream< _CharT, _Traits >`, 1450
 - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1483
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1767
 - `std::ios_base`, 2309
- showmanyc
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2132

- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2960
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2977
- `std::basic_filebuf< _CharT, _Traits >`, 1134
- `std::basic_streambuf< _CharT, _Traits >`, 1507
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1714
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3270
- showpoint
 - `std`, 651
 - `std::basic_fstream< _CharT, _Traits >`, 1190
 - `std::basic_ifstream< _CharT, _Traits >`, 1234
 - `std::basic_ios< _CharT, _Traits >`, 1258
 - `std::basic_iostream< _CharT, _Traits >`, 1305
 - `std::basic_istream< _CharT, _Traits >`, 1344
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1385
 - `std::basic_ofstream< _CharT, _Traits >`, 1418
 - `std::basic_ostream< _CharT, _Traits >`, 1450
 - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1483
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1768
 - `std::ios_base`, 2309
- showpos
 - `std`, 651
 - `std::basic_fstream< _CharT, _Traits >`, 1191
 - `std::basic_ifstream< _CharT, _Traits >`, 1234
 - `std::basic_ios< _CharT, _Traits >`, 1258
 - `std::basic_iostream< _CharT, _Traits >`, 1305
 - `std::basic_istream< _CharT, _Traits >`, 1345
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1385
 - `std::basic_ofstream< _CharT, _Traits >`, 1419
 - `std::basic_ostream< _CharT, _Traits >`, 1450
 - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1483
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1768
 - `std::ios_base`, 2309
- shrink_to_fit
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 886
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1691
 - `std::deque< _Tp, _Alloc >`, 2085
 - `std::vector< _Tp, _Alloc >`, 3252
- shuffle
 - Mutating, 38
- shuffle_order_engine
 - `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 2925, 2926
- signaling_NaN
 - `std::numeric_limits< _Tp >`, 2608
- simd, 3543
- sin
 - Complex Numbers, 190
 - math.h, 3636
- sinh
 - Complex Numbers, 191
 - math.h, 3636
- size
 - `__gnu_cxx::Temporary_buffer< _ForwardIterator, _Tp >`, 1048
 - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 886
 - `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`, 3005
 - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1587
 - Numeric Arrays, 236
 - `std`, 651, 652
 - `std::Temporary_buffer< _ForwardIterator, _Tp >`, 1050
 - `std::basic_string< _CharT, _Traits, _Alloc >`, 1691
 - `std::bitset< _Nb >`, 1820
 - `std::deque< _Tp, _Alloc >`, 2085
 - `std::list< _Tp, _Alloc >`, 2383
 - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2440
 - `std::match_results< _Bi_iter, _Alloc >`, 2453
 - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 2533
 - `std::multiset< _Key, _Compare, _Alloc >`, 2554
 - `std::priority_queue< _Tp, _Sequence, _Compare >`, 2787
 - `std::queue< _Tp, _Sequence >`, 2796
 - `std::set< _Key, _Compare, _Alloc >`, 2906
 - `std::stack< _Tp, _Sequence >`, 2943
 - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2116
 - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3146
 - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3172
 - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3196
 - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3221
 - `std::vector< _Tp, _Alloc >`, 3252
- size_fn_imps.hpp, 3571
- size_type
 - `__gnu_pbds::hash_prime_size_policy`, 2270
 - `__gnu_pbds::sample_range_hashing`, 2865
 - `__gnu_pbds::sample_resize_policy`, 2868
 - `__gnu_pbds::sample_resize_trigger`, 2870
 - `__gnu_pbds::sample_size_policy`, 2873
 - `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3073
 - `std::allocator_traits< _Alloc >`, 1064
 - `std::allocator_traits< allocator< _Tp > >`, 1070
 - `std::allocator_traits< allocator< void > >`, 1075

- std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1079](#)
- std::set< _Key, _Compare, _Alloc >, [2892](#)
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3126](#)
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3155](#)
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- skipws
 - std, [652](#)
 - std::basic_fstream< _CharT, _Traits >, [1191](#)
 - std::basic_ifstream< _CharT, _Traits >, [1234](#)
 - std::basic_ios< _CharT, _Traits >, [1258](#)
 - std::basic_iostream< _CharT, _Traits >, [1305](#)
 - std::basic_istream< _CharT, _Traits >, [1345](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1385](#)
 - std::basic_ofstream< _CharT, _Traits >, [1419](#)
 - std::basic_ostream< _CharT, _Traits >, [1450](#)
 - std::basic_ostreamstream< _CharT, _Traits, _Alloc >, [1484](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1768](#)
 - std::ios_base, [2309](#)
- sleep_for
 - std::this_thread, [733](#)
- sleep_until
 - std::this_thread, [733](#)
- slice
 - Numeric Arrays, [224](#), [225](#)
- slice_array
 - Numeric Arrays, [225](#)
- slice_array.h, [3401](#)
- slist, [3608](#)
- snnextc
 - __gnu_cxx::enc_filebuf< _CharT >, [2132](#)
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, [2960](#)
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, [2978](#)
 - std::basic_filebuf< _CharT, _Traits >, [1135](#)
 - std::basic_streambuf< _CharT, _Traits >, [1507](#)
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, [1714](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3271](#)
- sort
 - Sorting, [83](#)
 - std::forward_list< _Tp, _Alloc >, [2193](#)
 - std::list< _Tp, _Alloc >, [2383](#), [2384](#)
- sort.h, [3682](#)
- sort_heap
 - Heap, [96](#)
- sort_minimal_n
 - __gnu_parallel::_Settings, [1043](#)
- sort_mwms_oversampling
 - __gnu_parallel::_Settings, [1043](#)
- sort_qs_num_samples_preset
 - __gnu_parallel::_Settings, [1043](#)
- sort_qsb_base_case_maximal_n
 - __gnu_parallel::_Settings, [1043](#)
- Sorting, [62](#)
 - clamp, [65](#)
 - inplace_merge, [66](#)
 - is_sorted, [67](#)
 - is_sorted_until, [68](#)
 - lexicographical_compare, [69](#)
 - lexicographical_compare_three_way, [70](#)
 - max, [70](#), [71](#)
 - max_element, [71](#), [72](#)
 - merge, [72](#), [73](#)
 - min, [74](#), [75](#)
 - min_element, [75](#), [76](#)
 - minmax, [76](#), [77](#)
 - minmax_element, [77](#)
 - next_permutation, [78](#)
 - nth_element, [79](#)
 - partial_sort, [80](#)
 - partial_sort_copy, [81](#)
 - prev_permutation, [82](#)
 - sort, [83](#)
 - stable_sort, [84](#)
- source_location, [3693](#)
- span, [3693](#)
- specfun.h, [3401](#)
- sph_bessel
 - Mathematical Special Functions, [218](#)
 - TR1 Mathematical Special Functions, [256](#)
- sph_besself
 - Mathematical Special Functions, [219](#)
- sph_bessell
 - Mathematical Special Functions, [219](#)
- sph_legendre
 - Mathematical Special Functions, [219](#)
 - TR1 Mathematical Special Functions, [256](#)
- sph_legendref
 - Mathematical Special Functions, [220](#)
- sph_legendrel
 - Mathematical Special Functions, [220](#)
- sph_neumann
 - Mathematical Special Functions, [220](#)
 - TR1 Mathematical Special Functions, [256](#)
- sph_neumannf
 - Mathematical Special Functions, [221](#)
- sph_neumannl
 - Mathematical Special Functions, [221](#)
- splay_fnimps.hpp, [3586](#)
- splay_tree.hpp, [3586](#)

- splice
 - std::list< _Tp, _Alloc >, 2384, 2385
- splice_after
 - std::forward_list< _Tp, _Alloc >, 2193–2195
- split_fn_imps.hpp, 3583
- split_join_can_throw
 - __gnu_pbds::container_traits< Cntnr >, 1941
- split_join_fn_imps.hpp, 3564, 3565
- sputbackc
 - __gnu_cxx::enc_filebuf< _CharT >, 2132
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2960
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2978
 - std::basic_filebuf< _CharT, _Traits >, 1135
 - std::basic_streambuf< _CharT, _Traits >, 1507
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1714
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3271
- sputc
 - __gnu_cxx::enc_filebuf< _CharT >, 2133
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2961
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2978
 - std::basic_filebuf< _CharT, _Traits >, 1135
 - std::basic_streambuf< _CharT, _Traits >, 1508
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1715
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3271
- sputn
 - __gnu_cxx::enc_filebuf< _CharT >, 2133
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2961
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2979
 - std::basic_filebuf< _CharT, _Traits >, 1136
 - std::basic_streambuf< _CharT, _Traits >, 1508
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1715
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3272
- sqrt
 - Complex Numbers, 191
 - math.h, 3636
- sregex_token_iterator
 - Regular Expressions, 260
- ssize
 - std, 652, 653
- sso_string_base.h, 3609
- sstream, 3694
- sstream.tcc, 3403
- ssub_match
 - Regular Expressions, 260
- stable_partition
 - Mutating, 38
- stable_sort
 - Sorting, 84
- stack, 3694
 - std::stack< _Tp, _Sequence >, 2943
- standard_policies.hpp, 3587
- start
 - Numeric Arrays, 236, 237
- state
 - std::fpos< _StateT >, 2197, 2198
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3272
 - std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3282
- static_pointer_cast
 - Pointer Abstractions, 352
 - std, 653
- std, 476
 - _Construct, 598
 - _Destroy, 598, 599
 - _Destroy_n, 599
 - _find_if_not, 593
 - _find_if_not_n, 593
 - _gcd, 593
 - _gen_two_uniform_ints, 593
 - _inplace_stable_sort, 594
 - _lg, 594
 - _merge_adaptive, 594
 - _merge_without_buffer, 594
 - _move_median_to_first, 595
 - _move_merge, 595
 - _move_merge_adaptive, 595
 - _move_merge_adaptive_backward, 595
 - _partition, 595, 596
 - _ptr_rebind, 590
 - _reverse, 596
 - _rotate, 596, 597
 - _rotate_adaptive, 597
 - _sample, 597
 - _search_n_aux, 598
 - _stable_partition_adaptive, 598
 - _umap_traits, 590
 - _ummap_traits, 590
 - _umset_traits, 590
 - _uset_traits, 590
 - abs, 599
 - acosh, 599
 - advance, 599
 - begin, 600
 - boolalpha, 601
 - cbegin, 601
 - cend, 601
 - cerr, 657
 - chars_format, 592
 - cin, 657
 - clog, 657
 - compare_three_way_result_t, 590
 - const_pointer_cast, 601
 - cout, 657
 - crbegin, 602
 - crend, 602

data, 602, 603
dec, 603
default_sentinel, 657
defaultfloat, 603
denorm_absent, 592
denorm_indeterminate, 592
denorm_present, 592
destroying_delete, 657
distance, 603
dynamic_pointer_cast, 604
empty, 604
end, 605
endl, 606
ends, 606
fabs, 606
fixed, 606
float_denorm_style, 592
float_round_style, 592
flush, 606
from_chars, 607
generate_canonical, 607
get, 607–609
get_money, 609
get_new_handler, 609
get_temporary_buffer, 609
get_time, 610
getline, 610–612
hex, 612
hexfloat, 612
ignore, 657
index_sequence, 590
index_sequence_for, 590
inout_ptr, 612
internal, 613
io_errc, 592
isalnum, 613
isalpha, 613
isblank, 613
iscntrl, 614
isdigit, 614
isgraph, 614
islower, 614
isprint, 614
ispunct, 614
isspace, 615
isupper, 615
isxdigit, 615
iter_rvalue_reference_t, 591
left, 615
make_index_sequence, 591
make_integer_sequence, 591
new_handler, 591
noboolalpha, 615
noshowbase, 615
noshowpoint, 615
noshowpos, 616
noskipws, 616
nounitbuf, 616
nouppercase, 616
oct, 616
operator!=, 616, 617
operator<, 619–621
operator<<, 621–627
operator<=, 627, 628
operator<=>, 628–631
operator>, 636, 637
operator>>, 638–643
operator>=, 637, 638
operator+, 617–619
operator==, 631–636
operator&, 617
operator^, 643
operator|, 643
out_ptr, 644
projected, 591
put_money, 644
put_time, 644
rbegin, 645, 646
rend, 646, 647
replace_copy, 647
resetiosflags, 648
return_temporary_buffer, 648
right, 648
round_indeterminate, 592
round_to_nearest, 592
round_toward_infinity, 592
round_toward_neg_infinity, 592
round_toward_zero, 592
sample, 648
saturating_add, 649
saturating_cast, 649
saturating_div, 649
saturating_mul, 649
saturating_sub, 649
scientific, 649
search, 649
set_new_handler, 650
setbase, 650
setfill, 650
setiosflags, 650
setprecision, 651
setw, 651
showbase, 651
showpoint, 651
showpos, 651
size, 651, 652
skipws, 652
ssize, 652, 653

- static_pointer_cast, 653
- streamoff, 591
- streampos, 591
- streamsize, 591
- swap, 653–656
- tolower, 656
- toupper, 656
- u16streampos, 591
- u32streampos, 591
- unitbuf, 656
- uppercase, 656
- wcerr, 658
- wcin, 658
- wclog, 658
- wcout, 658
- ws, 657
- wstreampos, 592
- std::__basic_future<_Res>, 763
 - _M_get_result, 765
- std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>, 767
 - do_out, 768
 - in, 769
 - out, 770
 - unshift, 770
- std::__ctype_abstract_base<_CharT>, 774
 - char_type, 776
 - do_is, 777
 - do_narrow, 777, 778
 - do_scan_is, 779
 - do_scan_not, 779
 - do_tolower, 780
 - do_toupper, 781
 - do_widen, 781, 782
 - is, 782, 783
 - narrow, 783, 784
 - scan_is, 784
 - scan_not, 785
 - tolower, 785, 786
 - toupper, 786
 - widen, 787
- std::__cxx11::collate<_CharT>, 1902
 - ~collate, 1904
 - char_type, 1903
 - collate, 1903, 1904
 - compare, 1904
 - do_compare, 1905
 - do_hash, 1905
 - do_transform, 1906
 - hash, 1906
 - id, 1907
 - string_type, 1903
 - transform, 1906
- std::__cxx11::collate_byname<_CharT>, 1907
 - char_type, 1908
 - string_type, 1908
- std::__debug, 658
 - swap, 665
- std::__debug::bitset<_Nb>, 1809
- std::__debug::deque<_Tp, _Allocator>, 2061
- std::__debug::forward_list<_Tp, _Alloc>, 2175
- std::__debug::list<_Tp, _Allocator>, 2363
- std::__debug::map<_Key, _Tp, _Compare, _Allocator>, 2417
- std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>, 2512
- std::__debug::multiset<_Key, _Compare, _Allocator>, 2536
- std::__debug::set<_Key, _Compare, _Allocator>, 2886
- std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>, 3118
- std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>, 3147
- std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>, 3173
- std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>, 3196
- std::__debug::vector<_Tp, _Allocator>, 3230
 - vector, 3233
- std::__detail, 665
 - __from_chars_alnum, 668
 - __from_chars_pow2_base, 668
 - operator<<, 668, 669
 - operator>>, 669
- std::__detail::BracketMatcher<_TraitsT, __icase, __collate>, 893
- std::__detail::Compiler<_TraitsT>, 896
- std::__detail::Executor<_Bilter, _Alloc, _TraitsT>, 906
- std::__detail::List_node_base, 940
- std::__detail::List_node_header, 941
- std::__detail::Quoted_string<_String, _CharT>, 980
- std::__detail::Scanner<_CharT>, 1036
 - _TokenT, 1037
- std::__detail::StateSeq<_TraitsT>, 1046
- std::__fdlist::Iterator<_Const, _Ptr>, 921
 - operator==, 922
- std::__fdlist::Node<_ValPtr>, 961
- std::__fdlist::Node_base<_VoidPtr>, 963
- std::__is_fast_hash<_Hash>, 803
- std::__is_location_invariant<_Tp>, 804
- std::__new_allocator<_Tp>, 812
- std::__numeric_limits_base, 813
 - digits, 815
 - digits10, 815
 - has_denorm, 815
 - has_denorm_loss, 815
 - has_infinity, 816
 - has_quiet_NaN, 816

- has_signaling_NaN, 816
- is_bounded, 816
- is_exact, 816
- is_iec559, 816
- is_integer, 816
- is_modulo, 816
- is_signed, 816
- is_specialized, 816
- max_digits10, 817
- max_exponent, 817
- max_exponent10, 817
- min_exponent, 817
- min_exponent10, 817
- radix, 817
- round_style, 817
- tinyness_before, 817
- traps, 817
- std::__parallel, 669
 - search, 686
- std::__parallel::__CRandNumber< _MustBeInt >, 897
- std::__unspecified__::exception_ptr, 2155
- std::_Base_bitset< 0 >, 889
 - _M_w, 890
- std::_Base_bitset< 1 >, 890
- std::_Base_bitset< _Nw >, 888
 - _M_w, 889
- std::_Bind< _Signature >, 892
- std::_Bind_result< _Result, _Signature >, 892
- std::_Deque_base< _Tp, _Alloc >, 897
 - _M_initialize_map, 898
- std::_Deque_iterator< _Tp, _Ref, _Ptr >, 899
 - _M_set_node, 900
- std::_Function_base, 908
- std::_Fwd_list_base< _Tp, _Alloc >, 909
- std::_Fwd_list_const_iterator< _Tp >, 911
 - operator==, 912
- std::_Fwd_list_iterator< _Tp >, 912
 - operator==, 913
- std::_Fwd_list_node< _Tp >, 913
- std::_Fwd_list_node_base, 914
- std::_List_base< _Tp, _Alloc >, 935
- std::_List_const_iterator< _Tp >, 937
- std::_List_iterator< _Tp >, 937
- std::_List_node< _Tp >, 938
- std::_Not_fn< _Fn >, 970
- std::_Placeholder< _Num >, 972
- std::_Sp_ebo_helper< _Tp, false >, 1044
- std::_Sp_ebo_helper< _Tp, true >, 1044
- std::_Temporary_buffer< _ForwardIterator, _Tp >, 1048
 - _M_requested_size, 1050
 - _Temporary_buffer, 1049
 - begin, 1050
 - end, 1050
 - size, 1050
- std::_Vector_base< _Tp, _Alloc >, 1051
- std::add_const< _Tp >, 1052
- std::add_cv< _Tp >, 1053
- std::add_lvalue_reference< _Tp >, 1053
- std::add_pointer< _Tp >, 1053
- std::add_rvalue_reference< _Tp >, 1054
- std::add_volatile< _Tp >, 1054
- std::adopt_lock_t, 1054
- std::aligned_storage< _Len, _Align >, 1055
- std::aligned_union< _Len, _Types >, 1055
- std::alignment_of< _Tp >, 1056
- std::allocator< _Tp >, 1058
- std::allocator< void >, 1059
 - operator==, 1060
- std::allocator_traits< _Alloc >, 1061
 - allocate, 1065
 - allocate_at_least, 1066
 - allocator_type, 1063
 - const_pointer, 1063
 - const_void_pointer, 1063
 - construct, 1066
 - deallocate, 1066
 - destroy, 1067
 - difference_type, 1064
 - is_always_equal, 1064
 - max_size, 1067
 - pointer, 1064
 - propagate_on_container_copy_assignment, 1064
 - propagate_on_container_move_assignment, 1064
 - propagate_on_container_swap, 1064
 - select_on_container_copy_construction, 1067
 - size_type, 1064
 - value_type, 1065
 - void_pointer, 1065
- std::allocator_traits< allocator< _Tp > >, 1068
 - allocate, 1070
 - allocate_at_least, 1071
 - allocator_type, 1069
 - const_pointer, 1069
 - const_void_pointer, 1069
 - construct, 1071
 - deallocate, 1072
 - destroy, 1072
 - difference_type, 1069
 - is_always_equal, 1069
 - max_size, 1072
 - pointer, 1069
 - propagate_on_container_copy_assignment, 1069
 - propagate_on_container_move_assignment, 1070
 - propagate_on_container_swap, 1070
 - select_on_container_copy_construction, 1073
 - size_type, 1070
 - value_type, 1070
 - void_pointer, 1070

std::allocator_traits< allocator< void > >, 1073
 allocate, 1075
 allocator_type, 1074
 const_pointer, 1074
 const_void_pointer, 1074
 construct, 1075
 deallocate, 1076
 destroy, 1076
 difference_type, 1074
 is_always_equal, 1074
 max_size, 1076
 pointer, 1074
 propagate_on_container_copy_assignment, 1075
 propagate_on_container_move_assignment, 1075
 propagate_on_container_swap, 1075
 select_on_container_copy_construction, 1076
 size_type, 1075
 value_type, 1075
 void_pointer, 1075
 std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1077
 allocate, 1079
 allocate_at_least, 1080
 allocator_type, 1078
 const_pointer, 1078
 const_void_pointer, 1078
 construct, 1080
 deallocate, 1081
 destroy, 1081
 difference_type, 1078
 is_always_equal, 1078
 max_size, 1081
 pointer, 1078
 propagate_on_container_copy_assignment, 1078
 propagate_on_container_move_assignment, 1078
 propagate_on_container_swap, 1079
 select_on_container_copy_construction, 1082
 size_type, 1079
 value_type, 1079
 void_pointer, 1079
 std::array< _Tp, _Nm >, 1086
 std::assignable_from, 741
 std::atomic< _Tp >, 1088
 std::atomic< _Tp * >, 1089
 std::atomic< bool >, 1090
 std::atomic< char >, 1091
 std::atomic< char16_t >, 1092
 std::atomic< char32_t >, 1093
 std::atomic< int >, 1093
 std::atomic< long >, 1094
 std::atomic< long long >, 1094
 std::atomic< short >, 1095
 std::atomic< signed char >, 1096
 std::atomic< unsigned char >, 1096
 std::atomic< unsigned int >, 1097
 std::atomic< unsigned long >, 1097
 std::atomic< unsigned long long >, 1098
 std::atomic< unsigned short >, 1099
 std::atomic< wchar_t >, 1099
 std::atomic_flag, 1100
 std::auto_ptr< _Tp >, 1100
 ~auto_ptr, 1102
 auto_ptr, 1102, 1103
 element_type, 1101
 get, 1103
 operator->, 1103
 operator=, 1103, 1104
 operator*, 1103
 release, 1104
 reset, 1104
 std::auto_ptr_ref< _Tp1 >, 1105
 std::back_insert_iterator< _Container >, 1105
 back_insert_iterator, 1107
 container_type, 1107
 iterator_category, 1107
 operator++, 1107
 operator=, 1108
 operator*, 1107
 pointer, 1107
 reference, 1107
 value_type, 1107
 std::bad_alloc, 1108
 what, 1109
 std::bad_cast, 1110
 what, 1110
 std::bad_exception, 1111
 what, 1111
 std::bad_function_call, 1111
 what, 1112
 std::bad_typeid, 1113
 what, 1114
 std::bad_weak_ptr, 1114
 what, 1115
 std::basic_filebuf< _CharT, _Traits >, 1119
 _M_buf, 1139
 _M_buf_locale, 1139
 _M_buf_size, 1139
 _M_create_pback, 1123
 _M_destroy_pback, 1123
 _M_ext_buf, 1139
 _M_ext_buf_size, 1139
 _M_ext_next, 1139
 _M_in_beg, 1139
 _M_in_cur, 1139
 _M_in_end, 1140
 _M_mode, 1140
 _M_out_beg, 1140
 _M_out_cur, 1140

- [_M_out_end, 1140](#)
- [_M_pback, 1140](#)
- [_M_pback_cur_save, 1140](#)
- [_M_pback_end_save, 1141](#)
- [_M_pback_init, 1141](#)
- [_M_reading, 1141](#)
- [_M_set_buffer, 1123](#)
- [~basic_filebuf, 1123](#)
- [basic_filebuf, 1123](#)
- [close, 1124](#)
- [eback, 1124](#)
- [egptr, 1124](#)
- [epptr, 1125](#)
- [gbump, 1125](#)
- [getloc, 1125](#)
- [gptr, 1125](#)
- [imbue, 1126](#)
- [in_avail, 1126](#)
- [is_open, 1126](#)
- [open, 1127, 1128](#)
- [overflow, 1128](#)
- [pbackfail, 1129](#)
- [pbase, 1129](#)
- [pbump, 1130](#)
- [pptr, 1130](#)
- [pubimbue, 1130](#)
- [pubseekoff, 1131](#)
- [pubseekpos, 1131](#)
- [pubsetbuf, 1131](#)
- [pubsync, 1131](#)
- [sbumpc, 1132](#)
- [seekoff, 1132](#)
- [seekpos, 1132](#)
- [setbuf, 1132](#)
- [setg, 1133](#)
- [setp, 1133](#)
- [sgetc, 1134](#)
- [sgetn, 1134](#)
- [showmanyc, 1134](#)
- [snextc, 1135](#)
- [sputbackc, 1135](#)
- [sputc, 1135](#)
- [sputn, 1136](#)
- [sungetc, 1136](#)
- [sync, 1136](#)
- [uflow, 1137](#)
- [underflow, 1137](#)
- [xsgetn, 1137](#)
- [xspn, 1138](#)
- [std::basic_fstream<_CharT, _Traits >, 1141](#)
 - [_M_gcount, 1186](#)
 - [_M_getloc, 1151](#)
 - [~basic_fstream, 1151](#)
 - [adjustfield, 1186](#)
 - [app, 1187](#)
 - [ate, 1187](#)
 - [bad, 1151](#)
 - [badbit, 1187](#)
 - [basefield, 1187](#)
 - [basic_fstream, 1150, 1151](#)
 - [beg, 1187](#)
 - [binary, 1187](#)
 - [boolalpha, 1187](#)
 - [clear, 1152](#)
 - [close, 1152](#)
 - [copyfmt, 1152](#)
 - [cur, 1188](#)
 - [dec, 1188](#)
 - [end, 1188](#)
 - [eof, 1152](#)
 - [eofbit, 1188](#)
 - [event, 1150](#)
 - [event_callback, 1148](#)
 - [exceptions, 1153](#)
 - [fail, 1153](#)
 - [failbit, 1188](#)
 - [fill, 1154](#)
 - [fixed, 1189](#)
 - [flags, 1154](#)
 - [floatfield, 1189](#)
 - [flush, 1155](#)
 - [fmtflags, 1148](#)
 - [gcount, 1155](#)
 - [get, 1155–1157](#)
 - [getline, 1158](#)
 - [getloc, 1159](#)
 - [good, 1159](#)
 - [goodbit, 1189](#)
 - [hex, 1189](#)
 - [ignore, 1160](#)
 - [imbue, 1161](#)
 - [in, 1189](#)
 - [init, 1161](#)
 - [internal, 1189](#)
 - [iostate, 1149](#)
 - [is_open, 1162](#)
 - [iword, 1162](#)
 - [left, 1190](#)
 - [narrow, 1162](#)
 - [oct, 1190](#)
 - [open, 1163](#)
 - [openmode, 1149](#)
 - [operator bool, 1164](#)
 - [operator!, 1164](#)
 - [operator<<, 1164–1169](#)
 - [operator>>, 1169–1174](#)
 - [out, 1190](#)
 - [peek, 1175](#)

precision, 1175
put, 1175
putback, 1176
pword, 1176
rdbuf, 1177
rdstate, 1177
read, 1178
readsome, 1178
register_callback, 1179
right, 1190
scientific, 1190
seekdir, 1149
seekg, 1179, 1180
seekp, 1180, 1181
setf, 1181
setstate, 1182
showbase, 1190
showpoint, 1190
showpos, 1191
skipws, 1191
sync, 1182
sync_with_stdio, 1182
tellg, 1183
tellp, 1183
tie, 1183, 1184
trunc, 1191
unget, 1184
unitbuf, 1191
unsetf, 1184
uppercase, 1191
widen, 1185
width, 1185
write, 1186
xalloc, 1186
std::basic_ifstream<_CharT, _Traits >, 1194
 _M_gcount, 1230
 _M_getloc, 1202
 __num_put_type, 1199
 ~basic_ifstream, 1202
adjustfield, 1230
app, 1230
ate, 1230
bad, 1202
badbit, 1230
basefield, 1230
basic_ifstream, 1201, 1202
beg, 1230
binary, 1231
boolalpha, 1231
clear, 1203
close, 1203
copyfmt, 1203
cur, 1231
dec, 1231
end, 1231
eof, 1203
eofbit, 1231
event, 1201
event_callback, 1199
exceptions, 1204
fail, 1204
failbit, 1231
fill, 1205
fixed, 1232
flags, 1205
floatfield, 1232
fmtflags, 1199
gcount, 1206
get, 1206–1208
getline, 1209
getloc, 1210
good, 1210
goodbit, 1232
hex, 1232
ignore, 1210, 1211
imbue, 1211
in, 1232
init, 1212
internal, 1233
iostate, 1200
is_open, 1212
iword, 1212
left, 1233
narrow, 1213
oct, 1233
open, 1213, 1214
openmode, 1200
operator bool, 1214
operator!, 1214
operator>>, 1215–1220
out, 1233
peek, 1220
precision, 1220, 1221
putback, 1221
pword, 1221
rdbuf, 1222
rdstate, 1222
read, 1223
readsome, 1223
register_callback, 1224
right, 1233
scientific, 1233
seekdir, 1200
seekg, 1224, 1225
setf, 1225
setstate, 1226
showbase, 1234
showpoint, 1234

- showpos, [1234](#)
- skipws, [1234](#)
- sync, [1226](#)
- sync_with_stdio, [1227](#)
- tellg, [1227](#)
- tie, [1227](#)
- trunc, [1234](#)
- unget, [1228](#)
- unitbuf, [1234](#)
- unsetf, [1228](#)
- uppercase, [1234](#)
- widen, [1228](#)
- width, [1229](#)
- xalloc, [1229](#)
- std::basic_ios<_CharT, _Traits >, [1235](#)
 - _M_getloc, [1243](#)
 - __ctype_type, [1239](#)
 - __num_get_type, [1239](#)
 - __num_put_type, [1239](#)
 - ~basic_ios, [1242](#)
 - adjustfield, [1254](#)
 - app, [1254](#)
 - ate, [1254](#)
 - bad, [1243](#)
 - badbit, [1255](#)
 - basefield, [1255](#)
 - basic_ios, [1242](#), [1243](#)
 - beg, [1255](#)
 - binary, [1255](#)
 - boolalpha, [1255](#)
 - char_type, [1240](#)
 - clear, [1243](#)
 - copyfmt, [1244](#)
 - cur, [1255](#)
 - dec, [1255](#)
 - end, [1256](#)
 - eof, [1244](#)
 - eofbit, [1256](#)
 - event, [1242](#)
 - event_callback, [1240](#)
 - exceptions, [1244](#)
 - fail, [1245](#)
 - failbit, [1256](#)
 - fill, [1245](#)
 - fixed, [1256](#)
 - flags, [1246](#)
 - floatfield, [1256](#)
 - fmtflags, [1240](#)
 - getloc, [1246](#)
 - good, [1246](#)
 - goodbit, [1256](#)
 - hex, [1257](#)
 - imbue, [1247](#)
 - in, [1257](#)
 - init, [1247](#)
 - int_type, [1241](#)
 - internal, [1257](#)
 - iostate, [1241](#)
 - isword, [1247](#)
 - left, [1257](#)
 - narrow, [1248](#)
 - oct, [1257](#)
 - off_type, [1241](#)
 - openmode, [1241](#)
 - operator bool, [1248](#)
 - operator!, [1248](#)
 - out, [1258](#)
 - pos_type, [1242](#)
 - precision, [1248](#), [1249](#)
 - pword, [1249](#)
 - rdbuf, [1249](#), [1250](#)
 - rdstate, [1250](#)
 - register_callback, [1250](#)
 - right, [1258](#)
 - scientific, [1258](#)
 - seekdir, [1242](#)
 - setf, [1251](#)
 - setstate, [1251](#)
 - showbase, [1258](#)
 - showpoint, [1258](#)
 - showpos, [1258](#)
 - skipws, [1258](#)
 - sync_with_stdio, [1252](#)
 - tie, [1252](#)
 - traits_type, [1242](#)
 - trunc, [1259](#)
 - unitbuf, [1259](#)
 - unsetf, [1253](#)
 - uppercase, [1259](#)
 - widen, [1253](#)
 - width, [1253](#), [1254](#)
 - xalloc, [1254](#)
- std::basic_istream<_CharT, _Traits >, [1259](#)
 - _M_gcount, [1301](#)
 - _M_getloc, [1267](#)
 - ~basic_istream, [1267](#)
 - adjustfield, [1301](#)
 - app, [1301](#)
 - ate, [1301](#)
 - bad, [1267](#)
 - badbit, [1302](#)
 - basefield, [1302](#)
 - basic_istream, [1267](#)
 - beg, [1302](#)
 - binary, [1302](#)
 - boolalpha, [1302](#)
 - clear, [1268](#)
 - copyfmt, [1268](#)

cur, 1302
dec, 1302
end, 1303
eof, 1268
eofbit, 1303
event, 1267
event_callback, 1265
exceptions, 1268, 1269
fail, 1269
failbit, 1303
fill, 1269, 1270
fixed, 1303
flags, 1270
floatfield, 1303
flush, 1271
fmtflags, 1265
gcount, 1271
get, 1271–1273
getline, 1274
getloc, 1275
good, 1275
goodbit, 1303
hex, 1304
ignore, 1275, 1276
imbue, 1276
in, 1304
init, 1277
internal, 1304
iostate, 1266
iword, 1277
left, 1304
narrow, 1278
oct, 1304
openmode, 1266
operator bool, 1278
operator!, 1278
operator<<, 1278–1283
operator>>, 1284–1289
out, 1305
peek, 1289
precision, 1290
put, 1290
putback, 1290
pword, 1291
rdbuf, 1291, 1292
rdstate, 1292
read, 1292
readsome, 1293
register_callback, 1294
right, 1305
scientific, 1305
seekdir, 1266
seekg, 1294
seekp, 1295
setf, 1296
setstate, 1296
showbase, 1305
showpoint, 1305
showpos, 1305
skipws, 1305
sync, 1297
sync_with_stdio, 1297
tellg, 1297
tellp, 1298
tie, 1298
trunc, 1306
unget, 1299
unitbuf, 1306
unsetf, 1299
uppercase, 1306
widen, 1299
width, 1300
write, 1300
xalloc, 1301
std::basic_istream<_CharT, _Traits >, 1306
 _M_gcount, 1340
 _M_getloc, 1314
 __num_put_type, 1311
 ~basic_istream, 1314
adjustfield, 1340
app, 1341
ate, 1341
bad, 1314
badbit, 1341
basefield, 1341
basic_istream, 1314
beg, 1341
binary, 1341
boolalpha, 1341
clear, 1314
copyfmt, 1315
cur, 1342
dec, 1342
end, 1342
eof, 1315
eofbit, 1342
event, 1313
event_callback, 1311
exceptions, 1315
fail, 1316
failbit, 1342
fill, 1316
fixed, 1343
flags, 1317
floatfield, 1343
fmtflags, 1312
gcount, 1317
get, 1318–1320

- getline, [1320](#), [1321](#)
- getloc, [1321](#)
- good, [1322](#)
- goodbit, [1343](#)
- hex, [1343](#)
- ignore, [1322](#), [1323](#)
- imbue, [1323](#)
- in, [1343](#)
- init, [1324](#)
- internal, [1343](#)
- iostate, [1312](#)
- isword, [1324](#)
- left, [1344](#)
- narrow, [1324](#)
- oct, [1344](#)
- openmode, [1313](#)
- operator bool, [1325](#)
- operator!, [1325](#)
- operator>>, [1325–1330](#)
- out, [1344](#)
- peek, [1331](#)
- precision, [1331](#)
- putback, [1331](#)
- pword, [1332](#)
- rdbuf, [1332](#), [1333](#)
- rdstate, [1333](#)
- read, [1333](#)
- readsome, [1334](#)
- register_callback, [1335](#)
- right, [1344](#)
- scientific, [1344](#)
- seekdir, [1313](#)
- seekg, [1335](#)
- setf, [1336](#)
- setstate, [1336](#)
- showbase, [1344](#)
- showpoint, [1344](#)
- showpos, [1345](#)
- skipws, [1345](#)
- sync, [1337](#)
- sync_with_stdio, [1337](#)
- tellg, [1338](#)
- tie, [1338](#)
- trunc, [1345](#)
- unget, [1338](#)
- unitbuf, [1345](#)
- unsetf, [1339](#)
- uppercase, [1345](#)
- widen, [1339](#)
- width, [1340](#)
- xalloc, [1340](#)
- std::basic_istream< _CharT, _Traits >::sentry, [2882](#)
- operator bool, [2883](#)
- sentry, [2883](#)
- traits_type, [2883](#)
- std::basic_istream< _CharT, _Traits, _Alloc >, [1345](#)
- _M_gcount, [1381](#)
- _M_getloc, [1354](#)
- __num_put_type, [1351](#)
- ~basic_istream, [1354](#)
- adjustfield, [1381](#)
- app, [1381](#)
- ate, [1381](#)
- bad, [1355](#)
- badbit, [1382](#)
- basefield, [1382](#)
- basic_istream, [1354](#)
- beg, [1382](#)
- binary, [1382](#)
- boolalpha, [1382](#)
- clear, [1355](#)
- copyfmt, [1355](#)
- cur, [1382](#)
- dec, [1382](#)
- end, [1383](#)
- eof, [1356](#)
- eofbit, [1383](#)
- event, [1353](#)
- event_callback, [1351](#)
- exceptions, [1356](#)
- fail, [1357](#)
- failbit, [1383](#)
- fill, [1357](#)
- fixed, [1383](#)
- flags, [1357](#), [1358](#)
- floatfield, [1383](#)
- fmtflags, [1352](#)
- gcount, [1358](#)
- get, [1358–1360](#)
- getline, [1361](#)
- getloc, [1362](#)
- good, [1362](#)
- goodbit, [1383](#)
- hex, [1384](#)
- ignore, [1362](#), [1363](#)
- imbue, [1364](#)
- in, [1384](#)
- init, [1364](#)
- internal, [1384](#)
- iostate, [1352](#)
- isword, [1364](#)
- left, [1384](#)
- narrow, [1365](#)
- oct, [1384](#)
- openmode, [1353](#)
- operator bool, [1365](#)
- operator!, [1365](#)
- operator>>, [1366–1371](#)

out, 1385
peek, 1371
precision, 1371, 1372
putback, 1372
pword, 1372
rdbuf, 1373
rdstate, 1373
read, 1374
readsome, 1374
register_callback, 1375
right, 1385
scientific, 1385
seekdir, 1353
seekg, 1375, 1376
setf, 1376
setstate, 1377
showbase, 1385
showpoint, 1385
showpos, 1385
skipws, 1385
str, 1377
sync, 1378
sync_with_stdio, 1378
tellg, 1378
tie, 1379
trunc, 1386
unget, 1379
unitbuf, 1386
unsetf, 1380
uppercase, 1386
widen, 1380
width, 1380, 1381
xalloc, 1381
std::basic_ofstream<_CharT, _Traits >, 1386
 _M_getloc, 1394
 __num_get_type, 1391
 ~basic_ofstream, 1394
adjustfield, 1415
app, 1415
ate, 1415
bad, 1394
badbit, 1415
basefield, 1415
basic_ofstream, 1393, 1394
beg, 1415
binary, 1415
boolalpha, 1415
clear, 1395
close, 1395
copyfmt, 1395
cur, 1416
dec, 1416
end, 1416
eof, 1395
eofbit, 1416
event, 1393
event_callback, 1391
exceptions, 1396
fail, 1396
failbit, 1416
fill, 1397
fixed, 1417
flags, 1397
floatfield, 1417
flush, 1398
fmtflags, 1391
getloc, 1398
good, 1398
goodbit, 1417
hex, 1417
imbue, 1398
in, 1417
init, 1399
internal, 1417
iostate, 1392
is_open, 1399
iword, 1399
left, 1418
narrow, 1400
oct, 1418
open, 1400, 1401
openmode, 1392
operator bool, 1401
operator!, 1401
operator<<, 1402–1407
out, 1418
precision, 1407
put, 1407
pword, 1408
rdbuf, 1408, 1409
rdstate, 1409
register_callback, 1409
right, 1418
scientific, 1418
seekdir, 1392
seekp, 1410
setf, 1410, 1411
setstate, 1411
showbase, 1418
showpoint, 1418
showpos, 1419
skipws, 1419
sync_with_stdio, 1411
tellp, 1412
tie, 1412
trunc, 1419
unitbuf, 1419
unsetf, 1413

- uppercase, 1419
- widen, 1413
- width, 1413
- write, 1414
- xalloc, 1414
- std::basic_ostream< _CharT, _Traits >, 1419
 - _M_getloc, 1427
 - __num_get_type, 1424
 - ~basic_ostream, 1427
 - adjustfield, 1446
 - app, 1446
 - ate, 1446
 - bad, 1427
 - badbit, 1447
 - basefield, 1447
 - basic_ostream, 1427
 - beg, 1447
 - binary, 1447
 - boolalpha, 1447
 - clear, 1427
 - copyfmt, 1428
 - cur, 1447
 - dec, 1447
 - end, 1448
 - eof, 1428
 - eofbit, 1448
 - event, 1426
 - event_callback, 1424
 - exceptions, 1428
 - fail, 1429
 - failbit, 1448
 - fill, 1429
 - fixed, 1448
 - flags, 1430
 - floatfield, 1448
 - flush, 1430
 - fmtflags, 1425
 - getloc, 1431
 - good, 1431
 - goodbit, 1448
 - hex, 1449
 - imbue, 1431
 - in, 1449
 - init, 1431
 - internal, 1449
 - iostate, 1425
 - isword, 1432
 - left, 1449
 - narrow, 1432
 - oct, 1449
 - openmode, 1426
 - operator bool, 1433
 - operator!, 1433
 - operator<<, 1433–1438
 - out, 1450
 - precision, 1438, 1439
 - put, 1439
 - pword, 1439
 - rdbuf, 1440
 - rdstate, 1441
 - register_callback, 1441
 - right, 1450
 - scientific, 1450
 - seekdir, 1426
 - seekp, 1441, 1442
 - setf, 1442
 - setstate, 1443
 - showbase, 1450
 - showpoint, 1450
 - showpos, 1450
 - skipws, 1450
 - sync_with_stdio, 1443
 - tellp, 1443
 - tie, 1444
 - trunc, 1451
 - unitbuf, 1451
 - unsetf, 1444
 - uppercase, 1451
 - widen, 1444
 - width, 1445
 - write, 1445
 - xalloc, 1446
- std::basic_ostream< _CharT, _Traits >::sentry, 2884
 - ~sentry, 2884
 - operator bool, 2885
 - sentry, 2884
- std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1451
 - _M_getloc, 1460
 - __num_get_type, 1457
 - ~basic_ostringstream, 1460
 - adjustfield, 1479
 - app, 1479
 - ate, 1480
 - bad, 1460
 - badbit, 1480
 - basefield, 1480
 - basic_ostringstream, 1459
 - beg, 1480
 - binary, 1480
 - boolalpha, 1480
 - clear, 1460
 - copyfmt, 1461
 - cur, 1480
 - dec, 1481
 - end, 1481
 - eof, 1461
 - eofbit, 1481
 - event, 1459

event_callback, 1457
exceptions, 1461
fail, 1462
failbit, 1481
fill, 1462
fixed, 1481
flags, 1463
floatfield, 1481
flush, 1463
fmtflags, 1457
getloc, 1464
good, 1464
goodbit, 1482
hex, 1482
imbue, 1464
in, 1482
init, 1464
internal, 1482
iostate, 1458
iword, 1465
left, 1482
narrow, 1465
oct, 1483
openmode, 1458
operator bool, 1466
operator!, 1466
operator<<, 1466–1471
out, 1483
precision, 1471, 1472
put, 1472
pword, 1472
rdbuf, 1473
rdstate, 1473
register_callback, 1474
right, 1483
scientific, 1483
seekdir, 1458
seekp, 1474
setf, 1475
setstate, 1475
showbase, 1483
showpoint, 1483
showpos, 1483
skipws, 1484
str, 1476
sync_with_stdio, 1476
tellp, 1477
tie, 1477
trunc, 1484
unitbuf, 1484
unsetf, 1477
uppercase, 1484
widen, 1478
width, 1478
write, 1479
xalloc, 1479
std::basic_regex< _Ch_type, _Rx_traits >, 1484
 ~basic_regex, 1489
 assign, 1489–1491
 basic_regex, 1486–1488
 flags, 1492
 getloc, 1492
 imbue, 1492
 mark_count, 1492
 operator=, 1492, 1493
 swap, 1494
std::basic_streambuf< _CharT, _Traits >, 1494
 _M_buf_locale, 1511
 _M_in_beg, 1511
 _M_in_cur, 1511
 _M_in_end, 1511
 _M_out_beg, 1511
 _M_out_cur, 1512
 _M_out_end, 1512
 __streambuf_type, 1497
 ~basic_streambuf, 1498
 basic_streambuf, 1498
 char_type, 1497
 eback, 1498
 egptr, 1499
 epptr, 1499
 gbump, 1499
 getloc, 1500
 gptr, 1500
 imbue, 1500
 in_avail, 1501
 int_type, 1497
 off_type, 1497
 overflow, 1501
 pbackfail, 1501
 pbase, 1502
 pbump, 1502
 pos_type, 1498
 pptr, 1502
 pubimbue, 1503
 pubseekoff, 1503
 pubseekpos, 1503
 pubsetbuf, 1504
 pubsync, 1504
 sbumpc, 1504
 seekoff, 1504
 seekpos, 1505
 setbuf, 1505
 setg, 1505
 setp, 1506
 sgetc, 1506
 sgetn, 1506
 showmanyc, 1507

snextc, 1507
sputbackc, 1507
putc, 1508
sputn, 1508
sungetc, 1509
sync, 1509
traits_type, 1498
uflow, 1509
underflow, 1509
xsgetn, 1510
xsputn, 1510
std::basic_string<_CharT, _Traits, _Alloc >, 1589
 __resize_and_overwrite, 1606
 ~basic_string, 1602, 1606
 append, 1606–1611
 assign, 1612–1618
 at, 1618–1620
 back, 1620
 basic_string, 1599–1605
 begin, 1621
 c_str, 1621
 capacity, 1621, 1622
 cbegin, 1622
 cend, 1622
 clear, 1622
 compare, 1622–1628
 copy, 1628, 1629
 crbegin, 1629, 1630
 crend, 1630
 data, 1630, 1631
 empty, 1631
 end, 1631
 erase, 1632–1634
 find, 1634–1637
 find_first_not_of, 1638–1641
 find_first_of, 1641–1644
 find_last_not_of, 1644–1647
 find_last_of, 1648–1651
 front, 1651
 get_allocator, 1652
 insert, 1652–1661
 length, 1662
 max_size, 1662
 npos, 1694
 operator+=", 1662–1665
 operator=, 1665–1667
 operator[], 1668, 1669
 pop_back, 1669
 push_back, 1669, 1670
 rbegin, 1670
 rend, 1670, 1671
 replace, 1671–1684
 reserve, 1685, 1686
 resize, 1686, 1687
 rfind, 1687–1690
 shrink_to_fit, 1691
 size, 1691
 substr, 1692
 subview, 1693
 swap, 1694
std::basic_string_view<_CharT, _Traits >, 1695
std::basic_stringbuf<_CharT, _Traits, _Alloc >, 1701
 _M_buf_locale, 1719
 _M_in_beg, 1719
 _M_in_cur, 1719
 _M_in_end, 1719
 _M_mode, 1719
 _M_out_beg, 1719
 _M_out_cur, 1719
 _M_out_end, 1719
 basic_stringbuf, 1704, 1705
 eback, 1705
 egptr, 1705
 epptr, 1706
 gbump, 1706
 getloc, 1706
 gptr, 1707
 imbue, 1707
 in_avail, 1707
 overflow, 1708
 pbackfail, 1708
 pbase, 1709
 pbump, 1709
 pptr, 1709
 pubimbue, 1710
 pubseekoff, 1710
 pubseekpos, 1710
 pubsetbuf, 1711
 pubsync, 1711
 sbumpc, 1711
 seekoff, 1711
 seekpos, 1712
 setbuf, 1712
 setg, 1712
 setp, 1713
 sgetc, 1713
 sgetn, 1713
 showmanyc, 1714
 snextc, 1714
 sputbackc, 1714
 putc, 1715
 sputn, 1715
 str, 1716
 sungetc, 1716
 sync, 1716
 uflow, 1717
 underflow, 1717
 xsgetn, 1717

- xspn, 1718
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1720
 - _M_gcount, 1764
 - _M_getloc, 1729
 - ~basic_stringstream, 1729
 - adjustfield, 1764
 - app, 1764
 - ate, 1764
 - bad, 1730
 - badbit, 1764
 - basefield, 1764
 - basic_stringstream, 1729
 - beg, 1764
 - binary, 1764
 - boolalpha, 1765
 - clear, 1730
 - copyfmt, 1730
 - cur, 1765
 - dec, 1765
 - end, 1765
 - eof, 1731
 - eofbit, 1765
 - event, 1728
 - event_callback, 1727
 - exceptions, 1731
 - fail, 1732
 - failbit, 1765
 - fill, 1732
 - fixed, 1766
 - flags, 1732, 1733
 - floatfield, 1766
 - flush, 1733
 - fmtflags, 1727
 - gcount, 1733
 - get, 1733–1735
 - getline, 1736
 - getloc, 1737
 - good, 1737
 - goodbit, 1766
 - hex, 1766
 - ignore, 1738
 - imbue, 1739
 - in, 1766
 - init, 1739
 - internal, 1767
 - iostate, 1728
 - isword, 1740
 - left, 1767
 - narrow, 1740
 - oct, 1767
 - openmode, 1728
 - operator bool, 1740
 - operator!, 1741
 - operator<<, 1741–1746
 - operator>>, 1746–1751
 - out, 1767
 - peek, 1752
 - precision, 1752
 - put, 1752
 - putback, 1753
 - pwd, 1753
 - rdbuf, 1754
 - rdstate, 1754
 - read, 1755
 - readsome, 1755
 - register_callback, 1756
 - right, 1767
 - scientific, 1767
 - seekdir, 1728
 - seekg, 1756, 1757
 - seekp, 1757
 - setf, 1758
 - setstate, 1758
 - showbase, 1767
 - showpoint, 1768
 - showpos, 1768
 - skipws, 1768
 - str, 1759
 - sync, 1759
 - sync_with_stdio, 1760
 - tellg, 1760
 - tellp, 1760
 - tie, 1761
 - trunc, 1768
 - unget, 1761
 - unitbuf, 1768
 - unsetf, 1761
 - uppercase, 1768
 - widen, 1762
 - width, 1762
 - write, 1763
 - xalloc, 1763
- std::bernoulli_distribution, 1768
 - bernoulli_distribution, 1769
 - max, 1770
 - min, 1770
 - operator(), 1770
 - operator==, 1771
 - p, 1770
 - param, 1770
 - reset, 1771
 - result_type, 1769
- std::bernoulli_distribution::param_type, 2734
- std::bidirectional_iterator_tag, 1771
- std::binary_function< _Arg1, _Arg2, _Result >, 1784
 - first_argument_type, 1786
 - result_type, 1786
 - second_argument_type, 1786

- std::binary_negate< _Predicate >, 1795
 - first_argument_type, 1796
 - result_type, 1796
 - second_argument_type, 1796
- std::binder1st< _Operation >, 1796
 - argument_type, 1797
 - result_type, 1797
- std::binder2nd< _Operation >, 1798
 - argument_type, 1799
 - result_type, 1799
- std::binomial_distribution< _IntType >, 1799
 - max, 1800
 - min, 1800
 - operator<<, 1801
 - operator>>, 1802
 - operator(), 1800
 - operator==, 1802
 - p, 1801
 - param, 1801
 - reset, 1801
 - result_type, 1800
 - t, 1801
- std::binomial_distribution< _IntType >::param_type, 2735
- std::bitset< _Nb >, 1810
 - all, 1815
 - any, 1815
 - bitset, 1813, 1814
 - count, 1815
 - flip, 1815
 - none, 1816
 - operator<<, 1816
 - operator<=, 1816
 - operator>>, 1817
 - operator>=, 1817
 - operator==, 1817
 - operator&=, 1816
 - operator[], 1817
 - operator~, 1818
 - operator^=, 1818
 - operator|=, 1818
 - reset, 1819
 - set, 1819
 - size, 1820
 - test, 1820
 - to_string, 1820
 - to_ulong, 1820
- std::bitset< _Nb >::reference, 2831
- std::cauchy_distribution< _RealType >, 1824
 - max, 1825
 - min, 1825
 - operator(), 1825
 - operator==, 1826
 - param, 1826
 - reset, 1826
 - result_type, 1825
- std::cauchy_distribution< _RealType >::param_type, 2735
- std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >, 1843
- std::char_traits< _CharT >, 1842
- std::char_traits< char >, 1845
- std::char_traits< wchar_t >, 1847
- std::chi_squared_distribution< _RealType >, 1849
 - max, 1851
 - min, 1851
 - operator<<, 1852
 - operator>>, 1852
 - operator(), 1851
 - operator==, 1852
 - param, 1851
 - reset, 1851
 - result_type, 1851
- std::chi_squared_distribution< _RealType >::param_type, 2736
- std::chrono, 686
- std::chrono::duration< _Rep, _Period >, 2102
- std::chrono::duration_values< _Rep >, 2104
- std::chrono::gps_clock, 2234
- std::chrono::hh_mm_ss< _Duration >, 2276
- std::chrono::steady_clock, 2982
- std::chrono::system_clock, 3001
- std::chrono::tai_clock, 3002
- std::chrono::time_point< _Clock, _Dur >, 3040
- std::chrono::treat_as_floating_point< _Rep >, 3049
- std::chrono::tzdb_list, 3093
 - erase_after, 3094
 - front, 3094
- std::chrono::tzdb_list::const_iterator, 1922
- std::chrono::utc_clock, 3223
- std::codecvt< _InternT, _ExternT, _StateT >, 1853
 - do_always_noconv, 1854
 - do_encoding, 1854
 - do_in, 1855
 - do_length, 1855
 - do_max_length, 1855
 - do_out, 1855
 - do_unshift, 1855
 - in, 1856
 - out, 1856
 - unshift, 1857
- std::codecvt< _InternT, _ExternT, encoding_state >, 1858
 - do_always_noconv, 1860
 - do_encoding, 1860
 - do_in, 1860
 - do_length, 1860
 - do_max_length, 1860
 - do_out, 1861
 - do_unshift, 1861

- in, 1861, 1862
- out, 1863
- unshift, 1864, 1865
- std::codecvt< char, char, mbstate_t >, 1866
 - do_always_noconv, 1867
 - do_encoding, 1867
 - do_in, 1868
 - do_length, 1868
 - do_max_length, 1868
 - do_out, 1868
 - do_unshift, 1868
 - in, 1869
 - out, 1870, 1871
 - unshift, 1872
- std::codecvt< char16_t, char, mbstate_t >, 1873
 - do_always_noconv, 1875
 - do_encoding, 1875
 - do_in, 1875
 - do_length, 1875
 - do_max_length, 1875
 - do_out, 1875
 - do_unshift, 1876
 - in, 1876, 1877
 - out, 1878
 - unshift, 1879, 1880
- std::codecvt< char32_t, char, mbstate_t >, 1880
 - do_always_noconv, 1882
 - do_encoding, 1882
 - do_in, 1882
 - do_length, 1883
 - do_max_length, 1883
 - do_out, 1883
 - do_unshift, 1883
 - in, 1883, 1884
 - out, 1885, 1886
 - unshift, 1887
- std::codecvt< wchar_t, char, mbstate_t >, 1888
 - do_always_noconv, 1890
 - do_encoding, 1890
 - do_in, 1890
 - do_length, 1890
 - do_max_length, 1890
 - do_out, 1890
 - do_unshift, 1891
 - in, 1891, 1892
 - out, 1893
 - unshift, 1894, 1895
- std::codecvt_base, 1895
- std::codecvt_byname< _InternT, _ExternT, _StateT >, 1896
 - do_always_noconv, 1898
 - do_encoding, 1898
 - do_in, 1898
 - do_length, 1899
 - do_max_length, 1899
 - do_out, 1899
 - do_unshift, 1899
 - in, 1900
 - out, 1900
 - unshift, 1901
- std::common_iterator< _It, _Sent >, 1908
- std::common_reference_with, 741
- std::common_type< _Tp >, 1909
- std::common_type< chrono::duration< _Rep, _Period > >, 1909
- std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >, 1910
- std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >, 1910
- std::common_type< chrono::time_point< _Clock, _Duration > >, 1910
- std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >, 1910
- std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >, 1911
- std::common_with, 742
- std::compare_three_way_result< _Tp, _Up >, 1911
- std::complex< _Tp >, 1911
 - complex, 1912
 - explicit, 1913
 - operator+=, 1913
 - operator-=, 1913
 - operator/=, 1914
 - operator=, 1914
 - operator*=: 1913
 - value_type, 1912
- std::complex< double >, 1914
- std::complex< float >, 1915
- std::complex< long double >, 1916
- std::condition_variable, 1919
- std::condition_variable_any, 1919
- std::conditional< _Cond, _Iftrue, _Iffalse >, 1920
- std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1922
 - first_argument_type, 1923
 - result_type, 1923
 - second_argument_type, 1923
- std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1924
 - first_argument_type, 1924
 - result_type, 1924
 - second_argument_type, 1924
- std::const_mem_fun_ref_t< _Ret, _Tp >, 1925
 - argument_type, 1925
 - result_type, 1925
- std::const_mem_fun_t< _Ret, _Tp >, 1926
 - argument_type, 1926

result_type, 1926
 std::constructible_from, 742
 std::contiguous_iterator_tag, 1945
 std::convertible_to, 742
 std::copy_constructible, 743
 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV...
 noexcept(_Noex)>, 1946
 copyable_function, 1947, 1948
 operator bool, 1949
 operator(), 1949
 operator=, 1949
 operator==, 1950
 swap, 1950
 std::counted_iterator< _It >, 1950
 std::ctype< _CharT >, 1952
 do_is, 1954
 do_narrow, 1954, 1955
 do_scan_is, 1956
 do_scan_not, 1956
 do_tolower, 1957
 do_toupper, 1957, 1958
 do_widen, 1958, 1959
 id, 1965
 is, 1959, 1960
 narrow, 1960, 1961
 scan_is, 1961
 scan_not, 1962
 tolower, 1962, 1963
 toupper, 1963
 widen, 1964
 std::ctype< char >, 1965
 ~ctype, 1969
 char_type, 1968
 classic_table, 1969
 ctype, 1969
 do_is, 1969, 1970
 do_narrow, 1971, 1972
 do_scan_is, 1973
 do_scan_not, 1974
 do_tolower, 1975, 1976
 do_toupper, 1977, 1978
 do_widen, 1978, 1979
 id, 1995
 is, 1980–1982
 narrow, 1983–1985
 scan_is, 1986
 scan_not, 1987, 1988
 table, 1988
 table_size, 1996
 tolower, 1988–1990
 toupper, 1991–1993
 widen, 1993–1995
 std::ctype< wchar_t >, 1996
 ~ctype, 1998
 char_type, 1998
 ctype, 1998
 do_is, 1999
 do_narrow, 1999, 2000
 do_scan_is, 2000
 do_scan_not, 2001
 do_tolower, 2001, 2002
 do_toupper, 2002, 2003
 do_widen, 2003
 id, 2009
 is, 2004
 narrow, 2005
 scan_is, 2006
 scan_not, 2006
 tolower, 2007
 toupper, 2007, 2008
 widen, 2008, 2009
 std::ctype_base, 2009
 std::ctype_byname< _CharT >, 2010
 do_is, 2013
 do_narrow, 2013, 2014
 do_scan_is, 2015
 do_scan_not, 2015
 do_tolower, 2016
 do_toupper, 2016, 2017
 do_widen, 2017, 2018
 id, 2024
 is, 2018, 2019
 narrow, 2019, 2020
 scan_is, 2020
 scan_not, 2021
 tolower, 2021, 2022
 toupper, 2022
 widen, 2023
 std::ctype_byname< char >, 2024
 classic_table, 2027
 do_is, 2027, 2028
 do_narrow, 2029, 2030
 do_scan_is, 2031
 do_scan_not, 2031, 2032
 do_tolower, 2032–2034
 do_toupper, 2034, 2035
 do_widen, 2036, 2037
 id, 2047
 is, 2037, 2038
 narrow, 2039, 2040
 scan_is, 2041
 scan_not, 2042
 table, 2043
 table_size, 2047
 tolower, 2043, 2044
 toupper, 2044–2046
 widen, 2046, 2047
 std::decimal, 694

- decimal32_to_long_long, 703
- std::decimal::decimal128, 2049
 - decimal128, 2050
- std::decimal::decimal32, 2050
 - decimal32, 2052
- std::decimal::decimal64, 2052
 - decimal64, 2053
- std::default_delete< _Tp >, 2054
 - default_delete, 2055
 - operator(), 2055
- std::default_delete< _Tp[] >, 2055
 - default_delete, 2056
 - operator(), 2056
- std::default_initializable, 743
- std::default_sentinel_t, 2059
- std::defer_lock_t, 2060
- std::deque< _Tp, _Alloc >, 2063
 - _M_fill_initialize, 2071
 - _M_new_elements_at_back, 2071
 - _M_new_elements_at_front, 2071
 - _M_pop_back_aux, 2071
 - _M_pop_front_aux, 2071
 - _M_push_back_aux, 2071
 - _M_push_front_aux, 2072
 - _M_range_check, 2072
 - _M_range_initialize, 2072
 - _M_reallocate_map, 2073
 - _M_reserve_elements_at_back, 2073
 - _M_reserve_elements_at_front, 2073
 - _M_reserve_map_at_back, 2073
 - _M_reserve_map_at_front, 2073
 - ~deque, 2070
 - assign, 2074
 - at, 2075
 - back, 2076
 - begin, 2076
 - cbegin, 2076
 - cend, 2076
 - clear, 2076
 - crbegin, 2077
 - crend, 2077
 - deque, 2068–2070
 - emplace, 2077
 - empty, 2077
 - end, 2077, 2078
 - erase, 2078
 - front, 2078, 2079
 - get_allocator, 2079
 - insert, 2079–2081
 - max_size, 2081
 - operator=, 2081, 2082
 - operator[], 2082
 - pop_back, 2083
 - pop_front, 2083
 - push_back, 2083
 - push_front, 2083
 - rbegin, 2084
 - rend, 2084
 - resize, 2084
 - shrink_to_fit, 2085
 - size, 2085
 - swap, 2085
- std::derived_from, 743
- std::destroying_delete_t, 2085
- std::destructible, 743
- std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2091
 - base, 2093
 - discard, 2093
 - discard_block_engine, 2092, 2093
 - max, 2093
 - min, 2093
 - operator<<, 2094
 - operator>>, 2095
 - operator(), 2094
 - operator==, 2095
 - result_type, 2092
 - seed, 2094
- std::discrete_distribution< _IntType >, 2095
 - max, 2097
 - min, 2097
 - operator<<, 2098
 - operator>>, 2098
 - operator(), 2097
 - operator==, 2098
 - param, 2097
 - probabilities, 2097
 - reset, 2098
 - result_type, 2097
- std::discrete_distribution< _IntType >::param_type, 2736
- std::divides< _Tp >, 2099
 - first_argument_type, 2100
 - result_type, 2100
 - second_argument_type, 2100
- std::divides< void >, 2100
 - first_argument_type, 2101
 - result_type, 2101
 - second_argument_type, 2101
- std::domain_error, 2101
 - what, 2102
- std::enable_if< bool, _Tp >, 2119
- std::enable_shared_from_this< _Tp >, 2119
- std::encoding_state, 2140
- std::equal_to< _Tp >, 2145
 - first_argument_type, 2145
 - result_type, 2145
 - second_argument_type, 2145
- std::equal_to< void >, 2146

- first_argument_type, 2147
- result_type, 2147
- second_argument_type, 2147
- std::equivalence_relation, 744
- std::error_category, 2147
 - default_error_condition, 2148
 - equivalent, 2148
 - message, 2148
 - name, 2148
 - operator<=, 2149
 - operator==, 2149
- std::error_code, 2149
 - category, 2150
 - default_error_condition, 2150
 - error_code, 2150
 - message, 2150
 - operator bool, 2150
 - value, 2150
- std::error_condition, 2151
 - assign, 2152
 - category, 2152
 - clear, 2152
 - error_condition, 2151
 - message, 2152
 - operator bool, 2152
 - value, 2152
- std::exception, 2154
 - what, 2155
- std::exception_ptr, 2156
- std::experimental, 703
 - gcd, 714
 - get_deleter, 714
 - is_bind_expression_v, 715
 - is_placeholder_v, 715
 - lcm, 714
 - make_boyer_moore_horspool_searcher, 714
 - make_boyer_moore_searcher, 714
 - make_default_searcher, 714
 - make_ostream_joiner, 715
 - not_fn, 715
 - sample, 715
- std::experimental::filesystem::v1::basic_string_view<
 - _CharT, _Traits >, 1697
- std::experimental::filesystem::v1::filesystem_error, 2167
 - what, 2168
- std::experimental::filesystem::v1::path, 2750
- std::experimental::filesystem::v1::path::iterator, 2337
- std::experimental::filesystem::v1::space_info, 2935
- std::experimental::fundamentals_v1::any, 1083
 - ~any, 1085
 - any, 1084, 1085
 - clear, 1085
 - empty, 1085
 - operator=, 1085, 1086
 - swap, 1086
 - type, 1086
- std::experimental::fundamentals_v1::bad_any_cast, 1109
 - what, 1110
- std::experimental::fundamentals_v1::bad_optional_access,
 - 1112
 - what, 1113
- std::experimental::fundamentals_v1::basic_string_view<
 - _CharT, _Traits >, 1699
- std::experimental::fundamentals_v1::in_place_t, 2279
- std::experimental::fundamentals_v1::nullopt_t, 2575
- std::experimental::fundamentals_v1::optional<
 - _Tp >, 2696
- std::experimental::fundamentals_v2::ostream_joiner<
 - _DelimT, _CharT, _Traits >, 2700
- std::experimental::fundamentals_v2::owner_less< shared_ptr<
 - _Tp > >, 2715
 - first_argument_type, 2715
 - result_type, 2715
 - second_argument_type, 2715
- std::experimental::fundamentals_v2::owner_less< weak_ptr<
 - _Tp > >, 2717
 - first_argument_type, 2717
 - result_type, 2717
 - second_argument_type, 2718
- std::experimental::fundamentals_v2::propagate_const<
 - _Tp >, 2792
- std::exponential_distribution< _RealType >, 2156
 - exponential_distribution, 2158
 - lambda, 2158
 - max, 2158
 - min, 2158
 - operator(), 2158
 - operator==, 2159
 - param, 2158, 2159
 - reset, 2159
 - result_type, 2158
- std::exponential_distribution< _RealType >::param_type,
 - 2737
- std::extent< typename, _UInt >, 2159
- std::extreme_value_distribution< _RealType >, 2160
 - a, 2161
 - b, 2161
 - max, 2162
 - min, 2162
 - operator(), 2162
 - operator==, 2163
 - param, 2162
 - reset, 2162
 - result_type, 2161
- std::extreme_value_distribution< _RealType >::param_type,
 - 2737
- std::filesystem, 715
- std::filesystem::directory_entry, 2088

std::filesystem::directory_iterator, 2089
 std::filesystem::file_status, 2167
 std::filesystem::filesystem_error, 2169
 what, 2170
 std::filesystem::path, 2753
 format, 2755
 operator<<, 2756
 operator<=>, 2756
 operator>>, 2756
 operator/, 2756
 operator==, 2756
 std::filesystem::path::iterator, 2338
 std::filesystem::recursive_directory_iterator, 2827
 std::filesystem::space_info, 2935
 std::fisher_f_distribution<_RealType>, 2170
 max, 2172
 min, 2172
 operator<<, 2173
 operator>>, 2173
 operator(), 2172
 operator==, 2173
 param, 2172
 reset, 2172
 result_type, 2172
 std::fisher_f_distribution<_RealType>::param_type, 2738
 std::forward_iterator_tag, 2174
 std::forward_list<_Tp, _Alloc>, 2178
 ~forward_list, 2183
 assign, 2183, 2184
 before_begin, 2184
 begin, 2185
 cbefore_begin, 2185
 cbegin, 2185
 cend, 2185
 clear, 2185
 emplace_after, 2186
 emplace_front, 2186
 empty, 2186
 end, 2186, 2187
 erase_after, 2187
 forward_list, 2180–2183
 front, 2188
 get_allocator, 2188
 insert_after, 2188, 2189
 max_size, 2190
 merge, 2190
 operator=, 2190, 2191
 pop_front, 2191
 push_front, 2191
 remove, 2192
 remove_if, 2192
 resize, 2192, 2193
 reverse, 2193
 sort, 2193
 splice_after, 2193–2195
 swap, 2195
 unique, 2195, 2196
 std::fpos<_StateT>, 2196
 fpos, 2197
 operator streamoff, 2197
 operator+, 2197
 operator+=, 2197
 operator-, 2197
 operator-=, 2197
 state, 2197, 2198
 std::from_chars_result, 2199
 std::front_insert_iterator<_Container>, 2200
 container_type, 2201
 front_insert_iterator, 2201
 iterator_category, 2201
 operator++, 2201, 2202
 operator=, 2202
 operator*, 2201
 pointer, 2201
 reference, 2201
 value_type, 2201
 std::function<_Res(_ArgTypes...)>, 2202
 function, 2203, 2204
 operator bool, 2205
 operator(), 2205
 operator=, 2205–2207
 swap, 2207
 target, 2207, 2208
 target_type, 2208
 std::function_ref<_Res(_ArgTypes...) _GLIBCXX_MOF_CV
 noexcept(_Noex)>, 2208
 function_ref, 2209, 2210
 operator(), 2210
 std::future<_Res>, 2210
 _M_get_result, 2212
 future, 2212
 get, 2212
 std::future<_Res &>, 2212
 _M_get_result, 2213
 future, 2213
 get, 2213
 std::future<void>, 2214
 _M_get_result, 2215
 future, 2215
 get, 2215
 std::future_error, 2215
 what, 2216
 std::gamma_distribution<_RealType>, 2216
 alpha, 2218
 beta, 2218
 gamma_distribution, 2218
 max, 2218

- min, [2218](#)
- operator<=, [2219](#)
- operator>=, [2220](#)
- operator(), [2218](#), [2219](#)
- operator==, [2220](#)
- param, [2219](#)
- reset, [2219](#)
- result_type, [2218](#)
- std::gamma_distribution< _RealType >::param_type, [2738](#)
- std::geometric_distribution< _IntType >, [2220](#)
 - max, [2222](#)
 - min, [2222](#)
 - operator(), [2222](#)
 - operator==, [2223](#)
 - p, [2222](#)
 - param, [2222](#)
 - reset, [2222](#)
 - result_type, [2221](#)
- std::geometric_distribution< _IntType >::param_type, [2739](#)
- std::greater< _Tp >, [2235](#)
 - first_argument_type, [2235](#)
 - result_type, [2235](#)
 - second_argument_type, [2235](#)
- std::greater< void >, [2236](#)
 - first_argument_type, [2237](#)
 - result_type, [2237](#)
 - second_argument_type, [2237](#)
- std::greater_equal< _Tp >, [2237](#)
 - first_argument_type, [2238](#)
 - result_type, [2238](#)
 - second_argument_type, [2238](#)
- std::greater_equal< void >, [2238](#)
 - first_argument_type, [2239](#)
 - result_type, [2239](#)
 - second_argument_type, [2239](#)
- std::gslice, [2240](#)
- std::gslice_array< _Tp >, [2241](#)
 - operator<=, [2243](#)
 - operator>=, [2243](#)
 - operator+=, [2243](#)
 - operator=, [2243](#)
 - operator/=: [2243](#)
 - operator%=: [2242](#)
 - operator&=: [2242](#)
 - operator*=, [2243](#)
 - operator^=: [2243](#)
 - operator|=, [2243](#)
- std::has_virtual_destructor< _Tp >, [2244](#)
- std::hash< __debug::bitset< _Nb > >, [2244](#)
- std::hash< __debug::vector< bool, _Alloc > >, [2245](#)
- std::hash< __gnu_cxx::__u16vstring >, [2245](#)
- std::hash< __gnu_cxx::__u32vstring >, [2245](#)
- std::hash< __gnu_cxx::__vstring >, [2245](#)
- std::hash< __gnu_cxx::__wvstring >, [2246](#)
- std::hash< __gnu_cxx::throw_value_limit >, [2246](#)
 - argument_type, [2247](#)
 - result_type, [2247](#)
- std::hash< __gnu_cxx::throw_value_random >, [2247](#)
 - argument_type, [2248](#)
 - result_type, [2248](#)
- std::hash< __gnu_debug::basic_string< _CharT > >, [2248](#)
- std::hash< __shared_ptr< _Tp, _Lp > >, [2248](#)
- std::hash< _Tp >, [2244](#)
- std::hash< _Tp * >, [2249](#)
- std::hash< basic_string< char, char_traits< char >, _Alloc > >, [2249](#)
- std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >, [2249](#)
- std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >, [2250](#)
- std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >, [2250](#)
- std::hash< bool >, [2250](#)
- std::hash< char >, [2251](#)
- std::hash< char16_t >, [2251](#)
- std::hash< char32_t >, [2251](#)
- std::hash< double >, [2251](#)
- std::hash< error_code >, [2252](#)
- std::hash< error_condition >, [2252](#)
- std::hash< experimental::optional< _Tp > >, [2252](#)
- std::hash< experimental::shared_ptr< _Tp > >, [2253](#)
- std::hash< float >, [2253](#)
- std::hash< int >, [2253](#)
- std::hash< long >, [2254](#)
- std::hash< long double >, [2254](#)
- std::hash< long long >, [2254](#)
- std::hash< shared_ptr< _Tp > >, [2254](#)
- std::hash< short >, [2255](#)
- std::hash< signed char >, [2255](#)
- std::hash< thread::id >, [2255](#)
- std::hash< type_index >, [2255](#)
- std::hash< unique_ptr< _Tp, _Dp > >, [2256](#)
- std::hash< unsigned char >, [2256](#)
- std::hash< unsigned int >, [2256](#)
- std::hash< unsigned long >, [2257](#)
- std::hash< unsigned long long >, [2257](#)
- std::hash< unsigned short >, [2257](#)
- std::hash< wchar_t >, [2257](#)
- std::hash<::bitset< _Nb > >, [2258](#)
- std::hash<::vector< bool, _Alloc > >, [2258](#)
- std::identity, [2279](#)
- std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, [2279](#)
 - base, [2281](#)
 - discard, [2281](#)

- independent_bits_engine, 2280, 2281
- max, 2282
- min, 2282
- operator>>, 2283
- operator(), 2282
- operator==, 2283
- result_type, 2280
- seed, 2282
- std::indirect_array< _Tp >, 2283
 - operator<=, 2285
 - operator>=, 2286
 - operator+=, 2285
 - operator-=, 2285
 - operator/=: 2285
 - operator%=: 2285
 - operator&=: 2285
 - operator*=, 2285
 - operator^=: 2286
 - operator|=, 2286
- std::indirectly_comparable, 744
- std::indirectly_copyable, 744
- std::indirectly_movable, 744
- std::indirectly_readable, 744
- std::indirectly_swappable, 745
- std::indirectly_writable, 745
- std::initializer_list< _E >, 2286
- std::inout_ptr_t< _Smart, _Pointer, _Args >, 2287
- std::input_iterator_tag, 2287
- std::insert_iterator< _Container >, 2289
 - container_type, 2291
 - insert_iterator, 2291
 - iterator_category, 2291
 - operator++, 2291
 - operator=, 2292
 - operator*, 2291
 - pointer, 2291
 - reference, 2291
 - value_type, 2291
- std::integer_sequence< _Tp, _Idx >, 2292
- std::integral_constant< _Tp, __v >, 2293
- std::invalid_argument, 2293
 - what, 2294
- std::invocable, 745
- std::ios_base, 2294
 - _M_getloc, 2300
 - ~ios_base, 2300
 - adjustfield, 2305
 - app, 2305
 - ate, 2305
 - badbit, 2305
 - basefield, 2305
 - beg, 2305
 - binary, 2306
 - boolalpha, 2306
 - cur, 2306
 - dec, 2306
 - end, 2306
 - eofbit, 2306
 - event, 2299
 - event_callback, 2298
 - failbit, 2306
 - fixed, 2307
 - flags, 2300
 - floatfield, 2307
 - fmtflags, 2298
 - getloc, 2300
 - goodbit, 2307
 - hex, 2307
 - imbue, 2301
 - in, 2307
 - internal, 2308
 - iostate, 2299
 - isword, 2301
 - left, 2308
 - oct, 2308
 - openmode, 2299
 - out, 2308
 - precision, 2301, 2302
 - pword, 2302
 - register_callback, 2302
 - right, 2308
 - scientific, 2308
 - seekdir, 2299
 - setf, 2303
 - showbase, 2309
 - showpoint, 2309
 - showpos, 2309
 - skipws, 2309
 - sync_with_stdio, 2303
 - trunc, 2309
 - unitbuf, 2309
 - unsetf, 2304
 - uppercase, 2309
 - width, 2304
 - xalloc, 2304
- std::ios_base::failure, 2166
 - what, 2166
- std::is_abstract< _Tp >, 2309
- std::is_arithmetic< _Tp >, 2310
- std::is_array< _Tp >, 2310
- std::is_assignable< _Tp, _Up >, 2310
- std::is_base_of< _Base, _Derived >, 2310
- std::is_bind_expression< _Bind< _Signature > >, 2311
- std::is_bind_expression< _Bind_result< _Result, _Signature > >, 2311
- std::is_bind_expression< _Tp >, 2311
- std::is_bind_expression< const _Bind< _Signature > >, 2312

- `std::is_bind_expression< const _Bind_result< _Result, _Signature > >`, 2312
- `std::is_bind_expression< const volatile _Bind< _Signature > >`, 2312
- `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`, 2312
- `std::is_bind_expression< volatile _Bind< _Signature > >`, 2313
- `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`, 2313
- `std::is_class< _Tp >`, 2313
- `std::is_compound< _Tp >`, 2313
- `std::is_const< _Tp >`, 2314
- `std::is_constructible< _Tp, _Args >`, 2314
- `std::is_copy_assignable< _Tp >`, 2314
- `std::is_copy_constructible< _Tp >`, 2314
- `std::is_default_constructible< _Tp >`, 2314
- `std::is_destructible< _Tp >`, 2315
- `std::is_empty< _Tp >`, 2315
- `std::is_enum< _Tp >`, 2315
- `std::is_error_code_enum< _Tp >`, 2315
- `std::is_error_code_enum< future_errc >`, 2316
- `std::is_error_condition_enum< _Tp >`, 2316
- `std::is_floating_point< _Tp >`, 2316
- `std::is_function< _Tp >`, 2317
- `std::is_fundamental< _Tp >`, 2317
- `std::is_integral< _Tp >`, 2317
- `std::is_layout_compatible< _Tp, _Up >`, 2317
- `std::is_literal_type< _Tp >`, 2318
- `std::is_lvalue_reference< typename >`, 2318
- `std::is_member_function_pointer< _Tp >`, 2318
- `std::is_member_object_pointer< _Tp >`, 2319
- `std::is_member_pointer< _Tp >`, 2319
- `std::is_move_assignable< _Tp >`, 2319
- `std::is_move_constructible< _Tp >`, 2319
- `std::is_nothrow_assignable< _Tp, _Up >`, 2319
- `std::is_nothrow_constructible< _Tp, _Args >`, 2320
- `std::is_nothrow_copy_assignable< _Tp >`, 2320
- `std::is_nothrow_copy_constructible< _Tp >`, 2320
- `std::is_nothrow_default_constructible< _Tp >`, 2320
- `std::is_nothrow_destructible< _Tp >`, 2321
- `std::is_nothrow_move_assignable< _Tp >`, 2321
- `std::is_nothrow_move_constructible< _Tp >`, 2321
- `std::is_object< _Tp >`, 2321
- `std::is_placeholder< _Placeholder< _Num > >`, 2323
- `std::is_placeholder< _Tp >`, 2322
- `std::is_pod< _Tp >`, 2324
- `std::is_pointer< _Tp >`, 2324
- `std::is_pointer_interconvertible_base_of< _Base, _Derived >`, 2324
- `std::is_polymorphic< _Tp >`, 2324
- `std::is_reference< _Tp >`, 2325
- `std::is_rvalue_reference< typename >`, 2325
- `std::is_same< _Tp, _Up >`, 2325
- `std::is_scalar< _Tp >`, 2325
- `std::is_signed< _Tp >`, 2326
- `std::is_standard_layout< _Tp >`, 2326
- `std::is_trivial< _Tp >`, 2326
- `std::is_trivially_assignable< _Tp, _Up >`, 2326
- `std::is_trivially_constructible< _Tp, _Args >`, 2327
- `std::is_trivially_copy_assignable< _Tp >`, 2327
- `std::is_trivially_copy_constructible< _Tp >`, 2327
- `std::is_trivially_copyable< _Tp >`, 2327
- `std::is_trivially_default_constructible< _Tp >`, 2328
- `std::is_trivially_destructible< _Tp >`, 2328
- `std::is_trivially_move_assignable< _Tp >`, 2328
- `std::is_trivially_move_constructible< _Tp >`, 2328
- `std::is_union< _Tp >`, 2329
- `std::is_unsigned< _Tp >`, 2329
- `std::is_void< _Tp >`, 2329
- `std::is_volatile< _Tp >`, 2329
- `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 2330
 - `difference_type`, 2331
 - `istream_iterator`, 2331, 2332
 - `iterator_category`, 2331
 - `operator==`, 2332
 - `pointer`, 2331
 - `reference`, 2331
 - `value_type`, 2331
- `std::istreambuf_iterator< _CharT, _Traits >`, 2332
 - `char_type`, 2334
 - `difference_type`, 2334
 - `equal`, 2335
 - `int_type`, 2334
 - `istream_type`, 2334
 - `istreambuf_iterator`, 2335
 - `iterator_category`, 2334
 - `operator++`, 2335
 - `operator*`, 2335
 - `pointer`, 2334
 - `reference`, 2334
 - `streambuf_type`, 2334
 - `traits_type`, 2334
 - `value_type`, 2334
- `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`, 2338
 - `difference_type`, 2339
 - `iterator_category`, 2339
 - `pointer`, 2340
 - `reference`, 2340
 - `value_type`, 2340
- `std::iterator_traits< _Iterator >`, 2340
- `std::iterator_traits< _Tp * >`, 2340
- `std::length_error`, 2351
 - `what`, 2351
- `std::less< _Tp >`, 2352
 - `first_argument_type`, 2352
 - `result_type`, 2352

- second_argument_type, 2352
- std::less< void >, 2353
 - first_argument_type, 2354
 - result_type, 2354
 - second_argument_type, 2354
- std::less_equal< _Tp >, 2354
 - first_argument_type, 2355
 - result_type, 2355
 - second_argument_type, 2355
- std::less_equal< void >, 2355
 - first_argument_type, 2356
 - result_type, 2356
 - second_argument_type, 2356
- std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2357
 - discard, 2359
 - increment, 2362
 - linear_congruential_engine, 2359
 - max, 2359
 - min, 2360
 - modulus, 2362
 - multiplier, 2362
 - operator<=, 2361
 - operator>, 2361
 - operator(), 2360
 - operator==, 2361
 - result_type, 2359
 - seed, 2360
- std::list< _Tp, _Alloc >, 2365
 - _M_create_node, 2372
 - ~list, 2372
 - assign, 2372, 2373
 - back, 2373
 - begin, 2373, 2374
 - cbegin, 2374
 - cend, 2374
 - clear, 2374
 - crbegin, 2374
 - crend, 2374
 - emplace, 2374
 - empty, 2375
 - end, 2375
 - erase, 2375, 2376
 - front, 2376
 - get_allocator, 2376
 - insert, 2376–2378
 - list, 2370, 2371
 - max_size, 2379
 - merge, 2379
 - operator=, 2380
 - pop_back, 2380
 - pop_front, 2381
 - push_back, 2381
 - push_front, 2381
 - rbegin, 2381, 2382
 - remove, 2382
 - remove_if, 2382
 - rend, 2382
 - resize, 2383
 - reverse, 2383
 - size, 2383
 - sort, 2383, 2384
 - splice, 2384, 2385
 - swap, 2386
 - unique, 2386
- std::literals, 719
- std::literals::chrono_literals, 720
- std::locale, 2388
 - ~locale, 2393
 - all, 2396
 - category, 2390
 - classic, 2393
 - collate, 2396
 - combine, 2393
 - ctype, 2396
 - global, 2393
 - has_facet, 2395
 - locale, 2390–2392
 - messages, 2397
 - monetary, 2397
 - name, 2394
 - none, 2397
 - numeric, 2397
 - operator(), 2394
 - operator=, 2394
 - operator==, 2395
 - time, 2397
 - use_facet, 2396
- std::locale::facet, 2163
 - ~facet, 2165
 - facet, 2165
- std::locale::id, 2276
 - has_facet, 2277
 - id, 2277
 - use_facet, 2277
- std::lock_guard< _Mutex >, 2397
- std::logic_error, 2398
 - logic_error, 2399
 - what, 2399
- std::logical_and< _Tp >, 2399
 - first_argument_type, 2400
 - result_type, 2400
 - second_argument_type, 2400
- std::logical_and< void >, 2401
 - first_argument_type, 2401
 - result_type, 2401
 - second_argument_type, 2401
- std::logical_not< _Tp >, 2402

- argument_type, 2402
- result_type, 2402
- std::logical_not< void >, 2403
 - argument_type, 2403
 - result_type, 2403
- std::logical_or< _Tp >, 2404
 - first_argument_type, 2404
 - result_type, 2404
 - second_argument_type, 2404
- std::logical_or< void >, 2405
 - first_argument_type, 2405
 - result_type, 2405
 - second_argument_type, 2405
- std::lognormal_distribution< _RealType >, 2406
 - max, 2407
 - min, 2407
 - operator<=, 2408
 - operator>=, 2408
 - operator(), 2407
 - operator==, 2408
 - param, 2407
 - reset, 2408
 - result_type, 2407
- std::lognormal_distribution< _RealType >::param_type, 2739
- std::make_signed< _Tp >, 2416
- std::make_unsigned< _Tp >, 2416
- std::map< _Key, _Tp, _Compare, _Alloc >, 2419
 - ~map, 2426
 - at, 2426
 - begin, 2426, 2427
 - cbegin, 2427
 - cend, 2427
 - clear, 2427
 - contains, 2427
 - count, 2428
 - crbegin, 2428
 - crend, 2428
 - emplace, 2428
 - emplace_hint, 2429
 - empty, 2429
 - end, 2430
 - equal_range, 2430
 - erase, 2431, 2432
 - find, 2433
 - get_allocator, 2434
 - insert, 2434–2437
 - key_comp, 2437
 - lower_bound, 2437, 2438
 - map, 2423–2425
 - max_size, 2438
 - operator=, 2438, 2439
 - operator[], 2439
 - rbegin, 2439
 - rend, 2440
 - size, 2440
 - swap, 2440
 - upper_bound, 2440, 2441
 - value_comp, 2441
- std::mask_array< _Tp >, 2441
 - operator<=, 2443
 - operator>=, 2443
 - operator+=, 2443
 - operator-=, 2443
 - operator/=: 2443
 - operator%=: 2443
 - operator&=: 2443
 - operator*=, 2443
 - operator^=: 2444
 - operator|=, 2444
- std::match_results< _Bi_iter, _Alloc >, 2445
 - ~match_results, 2448
 - begin, 2449
 - cbegin, 2449
 - cend, 2449
 - empty, 2449
 - end, 2449
 - format, 2450
 - get_allocator, 2450
 - length, 2451
 - match_results, 2448
 - max_size, 2451
 - operator=, 2451
 - operator[], 2451
 - position, 2452
 - prefix, 2452
 - ready, 2452
 - size, 2453
 - str, 2453
 - suffix, 2453
 - swap, 2454
- std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2456
 - first_argument_type, 2457
 - result_type, 2457
 - second_argument_type, 2457
- std::mem_fun1_t< _Ret, _Tp, _Arg >, 2458
 - first_argument_type, 2458
 - result_type, 2458
 - second_argument_type, 2458
- std::mem_fun_ref_t< _Ret, _Tp >, 2459
 - argument_type, 2459
 - result_type, 2459
- std::mem_fun_t< _Ret, _Tp >, 2460
 - argument_type, 2460
 - result_type, 2460
- std::mergeable, 746
- std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,

- __f >, 2461
 - discard, 2464
 - max, 2464
 - mersenne_twister_engine, 2464
 - min, 2464
 - operator<<, 2465
 - operator>>, 2466
 - operator==, 2465
 - result_type, 2464
- std::messages< _CharT >, 2466
 - ~messages, 2469
 - char_type, 2468
 - do_get, 2469
 - id, 2469
 - messages, 2468, 2469
 - string_type, 2468
- std::messages_base, 2470
- std::messages_byname< _CharT >, 2470
 - do_get, 2472
 - id, 2472
- std::minus< _Tp >, 2472
 - first_argument_type, 2473
 - result_type, 2473
 - second_argument_type, 2473
- std::minus< void >, 2474
 - first_argument_type, 2474
 - result_type, 2474
 - second_argument_type, 2474
- std::modulus< _Tp >, 2475
 - first_argument_type, 2476
 - result_type, 2476
 - second_argument_type, 2476
- std::modulus< void >, 2476
 - first_argument_type, 2477
 - result_type, 2477
 - second_argument_type, 2477
- std::money_base, 2477
- std::money_get< _CharT, _InIter >, 2479
 - ~money_get, 2480
 - char_type, 2480
 - do_get, 2481
 - get, 2481, 2482
 - id, 2483
 - iter_type, 2480
 - money_get, 2480
 - string_type, 2480
- std::money_put< _CharT, _OutIter >, 2483
 - ~money_put, 2485
 - char_type, 2484
 - do_put, 2485, 2486
 - id, 2488
 - iter_type, 2484
 - money_put, 2485
 - put, 2486, 2487
 - string_type, 2484
- std::moneypunct< _CharT, _Intl >, 2488
 - ~moneypunct, 2491
 - char_type, 2490
 - curr_symbol, 2491
 - decimal_point, 2491
 - do_curr_symbol, 2492
 - do_decimal_point, 2492
 - do_frac_digits, 2492
 - do_grouping, 2492
 - do_neg_format, 2493
 - do_negative_sign, 2493
 - do_pos_format, 2493
 - do_positive_sign, 2494
 - do_thousands_sep, 2494
 - frac_digits, 2494
 - grouping, 2495
 - id, 2497
 - intl, 2497
 - moneypunct, 2490, 2491
 - neg_format, 2495
 - negative_sign, 2495
 - pos_format, 2496
 - positive_sign, 2496
 - string_type, 2490
 - thousands_sep, 2496
- std::moneypunct_byname< _CharT, _Intl >, 2497
 - curr_symbol, 2499
 - decimal_point, 2499
 - do_curr_symbol, 2499
 - do_decimal_point, 2499
 - do_frac_digits, 2500
 - do_grouping, 2500
 - do_neg_format, 2500
 - do_negative_sign, 2501
 - do_pos_format, 2501
 - do_positive_sign, 2501
 - do_thousands_sep, 2502
 - frac_digits, 2502
 - grouping, 2502
 - id, 2504
 - neg_format, 2502
 - negative_sign, 2503
 - pos_format, 2503
 - positive_sign, 2504
 - thousands_sep, 2504
- std::move_constructible, 746
- std::move_iterator< _Iterator >, 2506
- std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV
 - noexcept(_Noex)>, 2507
 - move_only_function, 2508, 2509
 - operator bool, 2510
 - operator(), 2510
 - operator=, 2510

- operator==, [2511](#)
- swap, [2510](#), [2511](#)
- std::move_sentinel< _Sent >, [2511](#)
- std::multimap< _Key, _Tp, _Compare, _Alloc >, [2514](#)
 - ~multimap, [2520](#)
 - begin, [2520](#)
 - cbegin, [2520](#)
 - cend, [2520](#)
 - clear, [2521](#)
 - contains, [2521](#)
 - count, [2521](#)
 - crbegin, [2522](#)
 - crend, [2522](#)
 - emplace, [2522](#)
 - emplace_hint, [2523](#)
 - empty, [2523](#)
 - end, [2523](#)
 - equal_range, [2524](#)
 - erase, [2524](#)–[2526](#)
 - find, [2526](#)
 - get_allocator, [2527](#)
 - insert, [2527](#)–[2530](#)
 - key_comp, [2530](#)
 - lower_bound, [2530](#), [2531](#)
 - max_size, [2531](#)
 - multimap, [2517](#)–[2519](#)
 - operator=, [2531](#), [2532](#)
 - rbegin, [2532](#)
 - rend, [2532](#)
 - size, [2533](#)
 - swap, [2533](#)
 - upper_bound, [2533](#)
 - value_comp, [2534](#)
- std::multiplies< _Tp >, [2534](#)
 - first_argument_type, [2535](#)
 - result_type, [2535](#)
 - second_argument_type, [2535](#)
- std::multiplies< void >, [2535](#)
 - first_argument_type, [2536](#)
 - result_type, [2536](#)
 - second_argument_type, [2536](#)
- std::multiset< _Key, _Compare, _Alloc >, [2538](#)
 - ~multiset, [2544](#)
 - begin, [2544](#)
 - cbegin, [2544](#)
 - cend, [2544](#)
 - clear, [2544](#)
 - contains, [2545](#)
 - count, [2545](#)
 - crbegin, [2546](#)
 - crend, [2546](#)
 - emplace, [2546](#)
 - emplace_hint, [2546](#)
 - empty, [2547](#)
 - end, [2547](#)
 - equal_range, [2547](#)
 - erase, [2548](#), [2549](#)
 - find, [2549](#)
 - get_allocator, [2550](#)
 - insert, [2550](#), [2551](#)
 - key_comp, [2552](#)
 - lower_bound, [2552](#)
 - max_size, [2553](#)
 - multiset, [2541](#)–[2544](#)
 - operator=, [2553](#)
 - rbegin, [2553](#)
 - rend, [2554](#)
 - size, [2554](#)
 - swap, [2554](#)
 - upper_bound, [2554](#), [2555](#)
 - value_comp, [2555](#)
- std::mutex, [2559](#)
- std::negate< _Tp >, [2559](#)
 - argument_type, [2560](#)
 - result_type, [2560](#)
- std::negate< void >, [2560](#)
 - argument_type, [2561](#)
 - result_type, [2561](#)
- std::negative_binomial_distribution< _IntType >, [2561](#)
 - k, [2562](#)
 - max, [2562](#)
 - min, [2562](#)
 - operator<=, [2563](#)
 - operator>, [2564](#)
 - operator(), [2562](#)
 - operator==, [2564](#)
 - p, [2563](#)
 - param, [2563](#)
 - reset, [2563](#)
 - result_type, [2562](#)
- std::negative_binomial_distribution< _IntType >::param_type, [2740](#)
- std::nested_exception, [2564](#)
 - nested_exception, [2565](#)
 - nested_ptr, [2565](#)
 - rethrow_nested, [2565](#)
- std::normal_distribution< _RealType >, [2567](#)
 - max, [2569](#)
 - mean, [2569](#)
 - min, [2569](#)
 - normal_distribution, [2569](#)
 - operator<=, [2570](#)
 - operator>, [2571](#)
 - operator(), [2569](#)
 - operator==, [2571](#)
 - param, [2570](#)
 - reset, [2570](#)
 - result_type, [2569](#)

stddev, 2570
 std::normal_distribution< _RealType >::param_type, 2740
 std::not_equal_to< _Tp >, 2571
 first_argument_type, 2572
 result_type, 2572
 second_argument_type, 2572
 std::not_equal_to< void >, 2573
 first_argument_type, 2574
 result_type, 2574
 second_argument_type, 2574
 std::num_get< _CharT, _InIter >, 2576
 ~num_get, 2578
 char_type, 2578
 do_get, 2578–2585
 get, 2585–2592
 id, 2593
 iter_type, 2578
 num_get, 2578
 std::num_put< _CharT, _OutIter >, 2593
 ~num_put, 2595
 char_type, 2595
 do_put, 2596–2599
 id, 2606
 iter_type, 2595
 num_put, 2595
 put, 2600–2605
 std::numeric_limits< _Tp >, 2606
 denorm_min, 2607
 digits, 2609
 digits10, 2609
 epsilon, 2607
 has_denorm, 2609
 has_denorm_loss, 2609
 has_infinity, 2609
 has_quiet_NaN, 2609
 has_signaling_NaN, 2609
 infinity, 2607
 is_bounded, 2609
 is_exact, 2609
 is_iec559, 2609
 is_integer, 2609
 is_modulo, 2610
 is_signed, 2610
 is_specialized, 2610
 lowest, 2608
 max, 2608
 max_digits10, 2610
 max_exponent, 2610
 max_exponent10, 2610
 min, 2608
 min_exponent, 2610
 min_exponent10, 2610
 quiet_NaN, 2608
 radix, 2610
 round_error, 2608
 round_style, 2611
 signaling_NaN, 2608
 tinyness_before, 2611
 traps, 2611
 std::numeric_limits< bool >, 2611
 digits, 2613
 digits10, 2613
 has_denorm, 2613
 has_denorm_loss, 2613
 has_infinity, 2613
 has_quiet_NaN, 2613
 has_signaling_NaN, 2613
 is_bounded, 2613
 is_exact, 2613
 is_iec559, 2613
 is_integer, 2613
 is_modulo, 2614
 is_signed, 2614
 is_specialized, 2614
 max_digits10, 2614
 max_exponent, 2614
 max_exponent10, 2614
 min_exponent, 2614
 min_exponent10, 2614
 radix, 2614
 round_style, 2614
 tinyness_before, 2615
 traps, 2615
 std::numeric_limits< char >, 2615
 digits, 2617
 digits10, 2617
 has_denorm, 2617
 has_denorm_loss, 2617
 has_infinity, 2617
 has_quiet_NaN, 2617
 has_signaling_NaN, 2617
 is_bounded, 2617
 is_exact, 2617
 is_iec559, 2617
 is_integer, 2617
 is_modulo, 2618
 is_signed, 2618
 is_specialized, 2618
 max_digits10, 2618
 max_exponent, 2618
 max_exponent10, 2618
 min_exponent, 2618
 min_exponent10, 2618
 radix, 2618
 round_style, 2618
 tinyness_before, 2619
 traps, 2619

`std::numeric_limits< char16_t >`, 2619

- `digits`, 2621
- `digits10`, 2621
- `has_denorm`, 2621
- `has_denorm_loss`, 2621
- `has_infinity`, 2621
- `has_quiet_NaN`, 2621
- `has_signaling_NaN`, 2621
- `is_bounded`, 2621
- `is_exact`, 2621
- `is_iec559`, 2621
- `is_integer`, 2621
- `is_modulo`, 2622
- `is_signed`, 2622
- `is_specialized`, 2622
- `max_digits10`, 2622
- `max_exponent`, 2622
- `max_exponent10`, 2622
- `min_exponent`, 2622
- `min_exponent10`, 2622
- `radix`, 2622
- `round_style`, 2622
- `tinyness_before`, 2623
- `traps`, 2623

`std::numeric_limits< char32_t >`, 2623

- `digits`, 2625
- `digits10`, 2625
- `has_denorm`, 2625
- `has_denorm_loss`, 2625
- `has_infinity`, 2625
- `has_quiet_NaN`, 2625
- `has_signaling_NaN`, 2625
- `is_bounded`, 2625
- `is_exact`, 2625
- `is_iec559`, 2625
- `is_integer`, 2625
- `is_modulo`, 2626
- `is_signed`, 2626
- `is_specialized`, 2626
- `max_digits10`, 2626
- `max_exponent`, 2626
- `max_exponent10`, 2626
- `min_exponent`, 2626
- `min_exponent10`, 2626
- `radix`, 2626
- `round_style`, 2626
- `tinyness_before`, 2627
- `traps`, 2627

`std::numeric_limits< double >`, 2627

- `digits`, 2629
- `digits10`, 2629
- `has_denorm`, 2629
- `has_denorm_loss`, 2629
- `has_infinity`, 2629

- `has_quiet_NaN`, 2629

- `has_signaling_NaN`, 2629
- `is_bounded`, 2629
- `is_exact`, 2629
- `is_iec559`, 2629
- `is_integer`, 2629
- `is_modulo`, 2630
- `is_signed`, 2630
- `is_specialized`, 2630
- `max_digits10`, 2630
- `max_exponent`, 2630
- `max_exponent10`, 2630
- `min_exponent`, 2630
- `min_exponent10`, 2630
- `radix`, 2630
- `round_style`, 2630
- `tinyness_before`, 2631
- `traps`, 2631

`std::numeric_limits< float >`, 2631

- `digits`, 2633
- `digits10`, 2633
- `has_denorm`, 2633
- `has_denorm_loss`, 2633
- `has_infinity`, 2633
- `has_quiet_NaN`, 2633
- `has_signaling_NaN`, 2633
- `is_bounded`, 2633
- `is_exact`, 2633
- `is_iec559`, 2633
- `is_integer`, 2633
- `is_modulo`, 2634
- `is_signed`, 2634
- `is_specialized`, 2634
- `max_digits10`, 2634
- `max_exponent`, 2634
- `max_exponent10`, 2634
- `min_exponent`, 2634
- `min_exponent10`, 2634
- `radix`, 2634
- `round_style`, 2634
- `tinyness_before`, 2635
- `traps`, 2635

`std::numeric_limits< int >`, 2635

- `digits`, 2637
- `digits10`, 2637
- `has_denorm`, 2637
- `has_denorm_loss`, 2637
- `has_infinity`, 2637
- `has_quiet_NaN`, 2637
- `has_signaling_NaN`, 2637
- `is_bounded`, 2637
- `is_exact`, 2637
- `is_iec559`, 2637
- `is_integer`, 2637

- is_modulo, [2638](#)
- is_signed, [2638](#)
- is_specialized, [2638](#)
- max_digits10, [2638](#)
- max_exponent, [2638](#)
- max_exponent10, [2638](#)
- min_exponent, [2638](#)
- min_exponent10, [2638](#)
- radix, [2638](#)
- round_style, [2638](#)
- tinyness_before, [2639](#)
- traps, [2639](#)
- std::numeric_limits< long >, [2639](#)
 - digits, [2641](#)
 - digits10, [2641](#)
 - has_denorm, [2641](#)
 - has_denorm_loss, [2641](#)
 - has_infinity, [2641](#)
 - has_quiet_NaN, [2641](#)
 - has_signaling_NaN, [2641](#)
 - is_bounded, [2641](#)
 - is_exact, [2641](#)
 - is_iec559, [2641](#)
 - is_integer, [2641](#)
 - is_modulo, [2642](#)
 - is_signed, [2642](#)
 - is_specialized, [2642](#)
 - max_digits10, [2642](#)
 - max_exponent, [2642](#)
 - max_exponent10, [2642](#)
 - min_exponent, [2642](#)
 - min_exponent10, [2642](#)
 - radix, [2642](#)
 - round_style, [2642](#)
 - tinyness_before, [2643](#)
 - traps, [2643](#)
- std::numeric_limits< long double >, [2643](#)
 - digits, [2645](#)
 - digits10, [2645](#)
 - has_denorm, [2645](#)
 - has_denorm_loss, [2645](#)
 - has_infinity, [2645](#)
 - has_quiet_NaN, [2645](#)
 - has_signaling_NaN, [2645](#)
 - is_bounded, [2645](#)
 - is_exact, [2645](#)
 - is_iec559, [2645](#)
 - is_integer, [2645](#)
 - is_modulo, [2646](#)
 - is_signed, [2646](#)
 - is_specialized, [2646](#)
 - max_digits10, [2646](#)
 - max_exponent, [2646](#)
 - max_exponent10, [2646](#)
- min_exponent, [2646](#)
- min_exponent10, [2646](#)
- radix, [2646](#)
- round_style, [2646](#)
- tinyness_before, [2647](#)
- traps, [2647](#)
- std::numeric_limits< long long >, [2647](#)
 - digits, [2649](#)
 - digits10, [2649](#)
 - has_denorm, [2649](#)
 - has_denorm_loss, [2649](#)
 - has_infinity, [2649](#)
 - has_quiet_NaN, [2649](#)
 - has_signaling_NaN, [2649](#)
 - is_bounded, [2649](#)
 - is_exact, [2649](#)
 - is_iec559, [2649](#)
 - is_integer, [2649](#)
 - is_modulo, [2650](#)
 - is_signed, [2650](#)
 - is_specialized, [2650](#)
 - max_digits10, [2650](#)
 - max_exponent, [2650](#)
 - max_exponent10, [2650](#)
 - min_exponent, [2650](#)
 - min_exponent10, [2650](#)
 - radix, [2650](#)
 - round_style, [2650](#)
 - tinyness_before, [2651](#)
 - traps, [2651](#)
- std::numeric_limits< short >, [2651](#)
 - digits, [2653](#)
 - digits10, [2653](#)
 - has_denorm, [2653](#)
 - has_denorm_loss, [2653](#)
 - has_infinity, [2653](#)
 - has_quiet_NaN, [2653](#)
 - has_signaling_NaN, [2653](#)
 - is_bounded, [2653](#)
 - is_exact, [2653](#)
 - is_iec559, [2653](#)
 - is_integer, [2653](#)
 - is_modulo, [2654](#)
 - is_signed, [2654](#)
 - is_specialized, [2654](#)
 - max_digits10, [2654](#)
 - max_exponent, [2654](#)
 - max_exponent10, [2654](#)
 - min_exponent, [2654](#)
 - min_exponent10, [2654](#)
 - radix, [2654](#)
 - round_style, [2654](#)
 - tinyness_before, [2655](#)
 - traps, [2655](#)

`std::numeric_limits< signed char >`, 2655

- `digits`, 2657
- `digits10`, 2657
- `has_denorm`, 2657
- `has_denorm_loss`, 2657
- `has_infinity`, 2657
- `has_quiet_NaN`, 2657
- `has_signaling_NaN`, 2657
- `is_bounded`, 2657
- `is_exact`, 2657
- `is_iec559`, 2657
- `is_integer`, 2657
- `is_modulo`, 2658
- `is_signed`, 2658
- `is_specialized`, 2658
- `max_digits10`, 2658
- `max_exponent`, 2658
- `max_exponent10`, 2658
- `min_exponent`, 2658
- `min_exponent10`, 2658
- `radix`, 2658
- `round_style`, 2658
- `tinyness_before`, 2659
- `traps`, 2659

`std::numeric_limits< unsigned char >`, 2659

- `digits`, 2661
- `digits10`, 2661
- `has_denorm`, 2661
- `has_denorm_loss`, 2661
- `has_infinity`, 2661
- `has_quiet_NaN`, 2661
- `has_signaling_NaN`, 2661
- `is_bounded`, 2661
- `is_exact`, 2661
- `is_iec559`, 2661
- `is_integer`, 2661
- `is_modulo`, 2662
- `is_signed`, 2662
- `is_specialized`, 2662
- `max_digits10`, 2662
- `max_exponent`, 2662
- `max_exponent10`, 2662
- `min_exponent`, 2662
- `min_exponent10`, 2662
- `radix`, 2662
- `round_style`, 2662
- `tinyness_before`, 2663
- `traps`, 2663

`std::numeric_limits< unsigned int >`, 2663

- `digits`, 2665
- `digits10`, 2665
- `has_denorm`, 2665
- `has_denorm_loss`, 2665
- `has_infinity`, 2665

- `has_quiet_NaN`, 2665

- `has_signaling_NaN`, 2665
- `is_bounded`, 2665
- `is_exact`, 2665
- `is_iec559`, 2665
- `is_integer`, 2665
- `is_modulo`, 2666
- `is_signed`, 2666
- `is_specialized`, 2666
- `max_digits10`, 2666
- `max_exponent`, 2666
- `max_exponent10`, 2666
- `min_exponent`, 2666
- `min_exponent10`, 2666
- `radix`, 2666
- `round_style`, 2666
- `tinyness_before`, 2667
- `traps`, 2667

`std::numeric_limits< unsigned long >`, 2667

- `digits`, 2669
- `digits10`, 2669
- `has_denorm`, 2669
- `has_denorm_loss`, 2669
- `has_infinity`, 2669
- `has_quiet_NaN`, 2669
- `has_signaling_NaN`, 2669
- `is_bounded`, 2669
- `is_exact`, 2669
- `is_iec559`, 2669
- `is_integer`, 2669
- `is_modulo`, 2670
- `is_signed`, 2670
- `is_specialized`, 2670
- `max_digits10`, 2670
- `max_exponent`, 2670
- `max_exponent10`, 2670
- `min_exponent`, 2670
- `min_exponent10`, 2670
- `radix`, 2670
- `round_style`, 2670
- `tinyness_before`, 2671
- `traps`, 2671

`std::numeric_limits< unsigned long long >`, 2671

- `digits`, 2673
- `digits10`, 2673
- `has_denorm`, 2673
- `has_denorm_loss`, 2673
- `has_infinity`, 2673
- `has_quiet_NaN`, 2673
- `has_signaling_NaN`, 2673
- `is_bounded`, 2673
- `is_exact`, 2673
- `is_iec559`, 2673
- `is_integer`, 2673

- is_modulo, [2674](#)
- is_signed, [2674](#)
- is_specialized, [2674](#)
- max_digits10, [2674](#)
- max_exponent, [2674](#)
- max_exponent10, [2674](#)
- min_exponent, [2674](#)
- min_exponent10, [2674](#)
- radix, [2674](#)
- round_style, [2674](#)
- tinyness_before, [2675](#)
- traps, [2675](#)
- std::numeric_limits< unsigned short >, [2675](#)
 - digits, [2677](#)
 - digits10, [2677](#)
 - has_denorm, [2677](#)
 - has_denorm_loss, [2677](#)
 - has_infinity, [2677](#)
 - has_quiet_NaN, [2677](#)
 - has_signaling_NaN, [2677](#)
 - is_bounded, [2677](#)
 - is_exact, [2677](#)
 - is_iec559, [2677](#)
 - is_integer, [2677](#)
 - is_modulo, [2678](#)
 - is_signed, [2678](#)
 - is_specialized, [2678](#)
 - max_digits10, [2678](#)
 - max_exponent, [2678](#)
 - max_exponent10, [2678](#)
 - min_exponent, [2678](#)
 - min_exponent10, [2678](#)
 - radix, [2678](#)
 - round_style, [2678](#)
 - tinyness_before, [2679](#)
 - traps, [2679](#)
- std::numeric_limits< wchar_t >, [2679](#)
 - digits, [2681](#)
 - digits10, [2681](#)
 - has_denorm, [2681](#)
 - has_denorm_loss, [2681](#)
 - has_infinity, [2681](#)
 - has_quiet_NaN, [2681](#)
 - has_signaling_NaN, [2681](#)
 - is_bounded, [2681](#)
 - is_exact, [2681](#)
 - is_iec559, [2681](#)
 - is_integer, [2681](#)
 - is_modulo, [2682](#)
 - is_signed, [2682](#)
 - is_specialized, [2682](#)
 - max_digits10, [2682](#)
 - max_exponent, [2682](#)
 - max_exponent10, [2682](#)
- min_exponent, [2682](#)
- min_exponent10, [2682](#)
- radix, [2682](#)
- round_style, [2682](#)
- tinyness_before, [2683](#)
- traps, [2683](#)
- std::num_punct< _CharT >, [2683](#)
 - ~num_punct, [2686](#)
 - char_type, [2685](#)
 - decimal_point, [2686](#)
 - do_decimal_point, [2686](#)
 - do_falsename, [2686](#)
 - do_grouping, [2686](#)
 - do_thousands_sep, [2687](#)
 - do_truename, [2687](#)
 - falsename, [2687](#)
 - grouping, [2687](#)
 - id, [2688](#)
 - num_punct, [2685](#)
 - string_type, [2685](#)
 - thousands_sep, [2688](#)
 - truename, [2688](#)
- std::num_punct_byname< _CharT >, [2689](#)
 - decimal_point, [2690](#)
 - do_decimal_point, [2690](#)
 - do_falsename, [2690](#)
 - do_grouping, [2691](#)
 - do_thousands_sep, [2691](#)
 - do_truename, [2691](#)
 - falsename, [2691](#)
 - grouping, [2692](#)
 - id, [2693](#)
 - thousands_sep, [2692](#)
 - truename, [2692](#)
- std::once_flag, [2695](#)
 - call_once, [2695](#)
 - once_flag, [2695](#)
 - operator=, [2695](#)
- std::ostream_iterator< _Tp, _CharT, _Traits >, [2697](#)
 - char_type, [2698](#)
 - difference_type, [2698](#)
 - iterator_category, [2698](#)
 - operator=, [2700](#)
 - ostream_iterator, [2699](#), [2700](#)
 - ostream_type, [2698](#)
 - pointer, [2699](#)
 - reference, [2699](#)
 - traits_type, [2699](#)
 - value_type, [2699](#)
- std::ostreambuf_iterator< _CharT, _Traits >, [2701](#)
 - char_type, [2702](#)
 - difference_type, [2702](#)
 - failed, [2703](#)
 - iterator_category, [2702](#)

- operator++, 2703
- operator=, 2704
- operator*, 2703
- ostream_type, 2702
- ostreambuf_iterator, 2703
- pointer, 2702
- reference, 2702
- streambuf_type, 2702
- traits_type, 2703
- value_type, 2703
- std::out_of_range, 2704
 - what, 2705
- std::out_ptr_t< _Smart, _Pointer, _Args >, 2705
- std::output_iterator_tag, 2706
- std::overflow_error, 2713
 - what, 2714
- std::owner_less< _Tp >, 2715
- std::owner_less< shared_ptr< _Tp > >, 2716
 - first_argument_type, 2716
 - result_type, 2716
 - second_argument_type, 2716
- std::owner_less< void >, 2716
 - first_argument_type, 2717
 - result_type, 2717
 - second_argument_type, 2717
- std::owner_less< weak_ptr< _Tp > >, 2718
 - first_argument_type, 2718
 - result_type, 2718
 - second_argument_type, 2718
- std::packaged_task< _Res(_ArgTypes...)>, 2719
- std::pair< _T1, _T2 >, 2724
 - explicit, 2727, 2728
 - first, 2729
 - first_type, 2727
 - operator=, 2728, 2729
 - pair, 2727
 - second, 2730
 - second_type, 2727
 - swap, 2729
- std::permutable, 746
- std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2756
 - discard, 2758
 - max, 2758
 - min, 2758
 - operator<<, 2759
 - operator>>, 2760
 - operator(), 2759
 - operator==, 2760
 - philox_engine, 2758
 - seed, 2759
 - set_counter, 2759
- std::piecewise_constant_distribution< _RealType >, 2760
 - densities, 2762
 - intervals, 2762
 - max, 2762
 - min, 2762
 - operator<<, 2763
 - operator>>, 2763
 - operator(), 2762
 - operator==, 2763
 - param, 2762, 2763
 - reset, 2763
 - result_type, 2762
- std::piecewise_constant_distribution< _RealType >::param_type, 2741
- std::piecewise_construct_t, 2764
- std::piecewise_linear_distribution< _RealType >, 2764
 - densities, 2766
 - intervals, 2766
 - max, 2766
 - min, 2766
 - operator<<, 2767
 - operator>>, 2767
 - operator(), 2766
 - operator==, 2767
 - param, 2766
 - reset, 2766
 - result_type, 2765
- std::piecewise_linear_distribution< _RealType >::param_type, 2742
- std::placeholders, 721
- std::plus< _Tp >, 2768
 - first_argument_type, 2768
 - operator(), 2769
 - result_type, 2768
 - second_argument_type, 2769
- std::pmr::memory_resource, 2461
- std::pmr::monotonic_buffer_resource, 2504
 - do_allocate, 2506
 - do_deallocate, 2506
 - do_is_equal, 2506
- std::pmr::polymorphic_allocator< _Tp >, 2780
- std::pmr::pool_options, 2781
 - max_blocks_per_chunk, 2781
- std::pmr::unsynchronized_pool_resource, 3222
 - do_allocate, 3223
 - do_deallocate, 3223
 - do_is_equal, 3223
- std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2770
 - first_argument_type, 2771
 - result_type, 2771
 - second_argument_type, 2771
- std::pointer_to_unary_function< _Arg, _Result >, 2771
 - argument_type, 2772
 - result_type, 2772
- std::pointer_traits< _Ptr >, 2772

- difference_type, 2773
- element_type, 2773
- pointer, 2773
- pointer_to, 2773
- rebind, 2773
- std::pointer_traits< _Tp * >, 2774
 - difference_type, 2774
 - element_type, 2774
 - pointer, 2774, 2775
 - pointer_to, 2775, 2776
 - rebind, 2775
- std::poisson_distribution< _IntType >, 2776
 - max, 2778
 - mean, 2778
 - min, 2778
 - operator<=, 2779
 - operator>=, 2779
 - operator(), 2778
 - operator==, 2779
 - param, 2778
 - reset, 2779
 - result_type, 2777
- std::poisson_distribution< _IntType >::param_type, 2742
- std::predicate, 746
- std::priority_queue< _Tp, _Sequence, _Compare >, 2783
 - empty, 2787
 - pop, 2787
 - priority_queue, 2785, 2786
 - push, 2787
 - size, 2787
 - top, 2787
- std::promise< _Res >, 2790
- std::promise< _Res & >, 2790
- std::promise< void >, 2791
- std::queue< _Tp, _Sequence >, 2793
 - back, 2795
 - c, 2796
 - empty, 2795
 - front, 2795
 - pop, 2796
 - push, 2796
 - queue, 2795
 - size, 2796
- std::random_access_iterator_tag, 2798
- std::random_device, 2799
 - result_type, 2800
- std::range_error, 2800
 - what, 2801
- std::ranges::bidirectional_range, 747
- std::ranges::borrowed_range, 747
- std::ranges::common_range, 747
- std::ranges::contiguous_range, 747
- std::ranges::dangling, 2048
- std::ranges::elements_of< _Range, _Alloc >, 2118
- std::ranges::empty_view< _Tp >, 2118
- std::ranges::equal_to, 2146
- std::ranges::forward_range, 747
- std::ranges::greater, 2236
- std::ranges::greater_equal, 2238
- std::ranges::input_range, 748
- std::ranges::less, 2353
- std::ranges::less_equal, 2355
- std::ranges::not_equal_to, 2573
- std::ranges::output_range, 748
- std::ranges::random_access_range, 748
- std::ranges::range, 748
- std::ranges::single_view< _Tp >, 2928
- std::ranges::sized_range, 749
- std::ranges::subrange< _It, _Sent, _Kind >, 2994
- std::ranges::view, 749
- std::ranges::view_base, 3257
- std::ranges::view_interface< _Derived >, 3257
- std::ranges::viewable_range, 749
- std::rank< _Tp >, 2807
- std::ratio< _Num, _Den >, 2808
- std::ratio_equal< _R1, _R2 >, 2808
- std::ratio_greater< _R1, _R2 >, 2809
- std::ratio_greater_equal< _R1, _R2 >, 2810
- std::ratio_less< _R1, _R2 >, 2811
- std::ratio_less_equal< _R1, _R2 >, 2812
- std::ratio_not_equal< _R1, _R2 >, 2812
- std::raw_storage_iterator< _OutputIterator, _Tp >, 2813
 - difference_type, 2814
 - iterator_category, 2814
 - pointer, 2815
 - reference, 2815
 - value_type, 2815
- std::recursive_mutex, 2829
- std::recursive_timed_mutex, 2830
- std::reference_wrapper< _Tp >, 2832
 - cref, 2833
 - ref, 2833
- std::regex_constants, 721
 - __multiline, 727
 - __polynomial, 727
 - awk, 728
 - basic, 728
 - collate, 728
 - ECMAScript, 728
 - egrep, 728
 - error_backref, 723
 - error_badbrace, 723
 - error_badrepeat, 724
 - error_brace, 724
 - error_brack, 724
 - error_collate, 724
 - error_complexity, 724
 - error_ctype, 724

- error_escape, [724](#)
- error_paren, [724](#)
- error_range, [724](#)
- error_space, [725](#)
- error_stack, [725](#)
- error_type, [723](#)
- extended, [728](#)
- format_default, [728](#)
- format_first_only, [729](#)
- format_no_copy, [729](#)
- format_sed, [729](#)
- grep, [729](#)
- icase, [729](#)
- match_any, [729](#)
- match_continuous, [729](#)
- match_default, [730](#)
- match_flag_type, [723](#)
- match_not_bol, [730](#)
- match_not_bow, [730](#)
- match_not_eol, [730](#)
- match_not_eow, [730](#)
- match_not_null, [730](#)
- match_prev_avail, [730](#)
- multiline, [730](#)
- nosubs, [731](#)
- operator&, [725](#)
- operator&=, [725](#)
- operator~, [727](#)
- operator^, [726](#)
- operator^=, [726](#)
- operator|, [726](#)
- operator|=, [727](#)
- optimize, [731](#)
- syntax_option_type, [723](#)
- std::regex_error, [2833](#)
 - code, [2835](#)
 - regex_error, [2834](#)
 - what, [2835](#)
- std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, [2835](#)
 - operator++, [2837](#)
 - operator->, [2837](#)
 - operator=, [2837](#)
 - operator==, [2837](#)
 - operator*, [2837](#)
 - regex_iterator, [2836](#)
- std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, [2838](#)
 - operator++, [2841](#)
 - operator->, [2841](#)
 - operator=, [2842](#)
 - operator==, [2842](#)
 - operator*, [2841](#)
 - regex_token_iterator, [2839–2841](#)
- std::regex_traits< _Ch_type >, [2842](#)
 - getloc, [2843](#)
 - imbue, [2843](#)
 - isctype, [2844](#)
 - length, [2844](#)
 - lookup_classname, [2845](#)
 - lookup_collatename, [2846](#)
 - regex_traits, [2843](#)
 - transform, [2846](#)
 - transform_primary, [2847](#)
 - translate, [2847](#)
 - translate_nocase, [2847](#)
 - value, [2848](#)
- std::regular_invocable, [749](#)
- std::rel_ops, [731](#)
 - operator!=, [731](#)
 - operator<=, [732](#)
 - operator>, [732](#)
 - operator>=, [732](#)
- std::relation, [750](#)
- std::remove_all_extents< _Tp >, [2848](#)
- std::remove_const< _Tp >, [2849](#)
- std::remove_cv< _Tp >, [2849](#)
- std::remove_extent< _Tp >, [2849](#)
- std::remove_pointer< _Tp >, [2849](#)
- std::remove_reference< _Tp >, [2850](#)
- std::remove_volatile< _Tp >, [2850](#)
- std::result_of< _Signature >, [2852](#)
- std::reverse_iterator< _Iterator >, [2852](#)
 - base, [2855](#)
 - operator+, [2855](#)
 - operator++, [2855](#)
 - operator+=, [2856](#)
 - operator-, [2856](#)
 - operator->, [2857](#)
 - operator--, [2856](#)
 - operator=, [2856](#)
 - operator[], [2857](#)
 - operator*, [2855](#)
 - reverse_iterator, [2854](#)
- std::runtime_error, [2862](#)
 - runtime_error, [2863](#)
 - what, [2863](#)
- std::same_as, [750](#)
- std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >, [2878](#)
- std::seed_seq, [2879](#)
 - result_type, [2880](#)
 - seed_seq, [2880](#)
- std::set< _Key, _Compare, _Alloc >, [2888](#)
 - ~set, [2895](#)
 - allocator_type, [2891](#)
 - begin, [2896](#)
 - cbegin, [2896](#)

- cend, 2896
- clear, 2896
- const_iterator, 2891
- const_pointer, 2891
- const_reference, 2891
- const_reverse_iterator, 2891
- contains, 2896
- count, 2897
- crbegin, 2897
- crend, 2897
- difference_type, 2891
- emplace, 2897
- emplace_hint, 2898
- empty, 2898
- end, 2898
- equal_range, 2899
- erase, 2899, 2900
- find, 2901
- get_allocator, 2901
- insert, 2902, 2903
- iterator, 2891
- key_comp, 2904
- key_compare, 2891
- key_type, 2892
- lower_bound, 2904
- max_size, 2905
- operator=, 2905
- pointer, 2892
- rbegin, 2905
- reference, 2892
- rend, 2906
- reverse_iterator, 2892
- set, 2893–2895
- size, 2906
- size_type, 2892
- swap, 2906
- upper_bound, 2906, 2907
- value_comp, 2907
- value_compare, 2892
- value_type, 2892
- std::shared_future< _Res >, 2907
 - _M_get_result, 2909
 - get, 2909
 - shared_future, 2908
- std::shared_future< _Res & >, 2909
 - _M_get_result, 2910
 - get, 2910
 - shared_future, 2910
- std::shared_future< void >, 2911
 - _M_get_result, 2912
 - shared_future, 2912
- std::shared_lock< _Mutex >, 2912
- std::shared_ptr< _Tp >, 2913
 - element_type, 2916
 - get, 2922
 - operator bool, 2922
 - owner_before, 2922
 - shared_ptr, 2916–2922
 - swap, 2922
 - unique, 2923
 - use_count, 2923
- std::shared_timed_mutex, 2923
- std::shuffle_order_engine< _RandomNumberEngine, __k >, 2924
 - base, 2926
 - discard, 2926
 - max, 2926
 - min, 2926
 - operator<=, 2927
 - operator>=, 2928
 - operator(), 2926
 - operator==, 2927
 - result_type, 2925
 - seed, 2927
 - shuffle_order_engine, 2925, 2926
- std::simd::__vec_builtin, 750
- std::simd::__vec_builtin_of, 750
- std::simd::rebind< _Tp, _Mp, _Traits >, 2826
- std::simd::rebind< _Tp, _Vp, _Traits >, 2827
- std::slice, 2930
 - operator==, 2930
- std::slice_array< _Tp >, 2930
 - operator<=, 2932
 - operator>=, 2932
 - operator+=, 2932
 - operator-=, 2932
 - operator/=, 2932
 - operator%=, 2932
 - operator&=, 2932
 - operator*=, 2932
 - operator^=, 2933
 - operator|=, 2933
- std::sortable, 751
- std::stack< _Tp, _Sequence >, 2941
 - empty, 2943
 - pop, 2943
 - push, 2943
 - size, 2943
 - stack, 2943
 - top, 2943, 2944
- std::strict_weak_order, 751
- std::student_t_distribution< _RealType >, 2987
 - max, 2988
 - min, 2988
 - operator<=, 2989
 - operator>=, 2990
 - operator(), 2988
 - operator==, 2989

- param, 2989
- reset, 2989
- result_type, 2988
- std::student_t_distribution< _RealType >::param_type, 2743
- std::sub_match< _Bilter >, 2990
 - compare, 2992
 - length, 2993
 - operator string_type, 2993
 - str, 2993
 - swap, 2993
- std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2995
 - discard, 2997
 - max, 2997
 - min, 2997
 - operator<<, 2998
 - operator>>, 2998
 - operator(), 2997
 - operator==, 2998
 - result_type, 2996
 - seed, 2997
 - subtract_with_carry_engine, 2996
- std::system_error, 3002
 - what, 3002
- std::this_thread, 733
 - get_id, 733
 - sleep_for, 733
 - sleep_until, 733
 - yield, 734
- std::thread, 3008
 - native_handle, 3009
- std::thread::id, 2278
- std::time_base, 3016
- std::time_get< _CharT, _InIter >, 3017
 - ~time_get, 3019
 - char_type, 3019
 - date_order, 3019
 - do_date_order, 3020
 - do_get, 3020
 - do_get_date, 3021
 - do_get_monthname, 3021
 - do_get_time, 3022
 - do_get_weekday, 3023
 - do_get_year, 3023
 - get, 3024, 3025
 - get_date, 3025
 - get_monthname, 3026
 - get_time, 3027
 - get_weekday, 3027
 - get_year, 3028
 - id, 3029
 - iter_type, 3019
 - time_get, 3019
- std::time_get_byname< _CharT, _InIter >, 3029
 - date_order, 3031
 - do_date_order, 3031
 - do_get, 3031
 - do_get_date, 3032
 - do_get_monthname, 3032
 - do_get_time, 3033
 - do_get_weekday, 3034
 - do_get_year, 3034
 - get, 3035, 3036
 - get_date, 3037
 - get_monthname, 3037
 - get_time, 3038
 - get_weekday, 3038
 - get_year, 3039
 - id, 3040
- std::time_put< _CharT, _OutIter >, 3041
 - ~time_put, 3042
 - char_type, 3042
 - do_put, 3043
 - id, 3045
 - iter_type, 3042
 - put, 3043, 3044
 - time_put, 3042
- std::time_put_byname< _CharT, _OutIter >, 3045
 - do_put, 3046
 - id, 3048
 - put, 3047
- std::timed_mutex, 3048
- std::to_chars_result, 3049
- std::tr1, 734
 - arg, 737
 - conf_hyperg, 737
 - conf_hypergfg, 737
 - conf_hypergl, 738
 - hyperg, 738
 - hypergfg, 738
 - hypergl, 739
 - norm, 739
 - polar, 739
- std::tr1::__detail, 739
- std::tr2, 739
- std::tr2::__detail, 741
- std::tr2::__dynamic_bitset_base< _WordT, _Alloc >, 788
 - _M_w, 790
- std::tr2::__reflection_typelist< _Elements >, 826
- std::tr2::__reflection_typelist< _First, _Rest... >, 826
- std::tr2::__reflection_typelist<>, 827
- std::tr2::bases< _Tp >, 1118
- std::tr2::bool_set, 1821
 - bool_set, 1822
 - equals, 1822
 - is_emptyset, 1822
 - is_indeterminate, 1822

is_singleton, 1822
 operator bool, 1822
 std::tr2::direct_bases< _Tp >, 2086
 std::tr2::dynamic_bitset< _WordT, _Alloc >, 2104
 all, 2109
 any, 2109
 append, 2109
 clear, 2109
 count, 2109
 dynamic_bitset, 2107–2109
 empty, 2110
 find_first, 2110
 find_next, 2110
 flip, 2110
 get_allocator, 2111
 max_size, 2111
 none, 2111
 num_blocks, 2111
 operator<<, 2112
 operator<=, 2112
 operator>>, 2113
 operator>=, 2113
 operator=, 2112
 operator=, 2113
 operator&=, 2111, 2112
 operator[], 2113, 2114
 operator~, 2114
 operator^=, 2114
 operator|=, 2114
 push_back, 2115
 reset, 2115
 resize, 2115
 set, 2115
 size, 2116
 swap, 2116
 test, 2116
 to_string, 2117
 to_ullong, 2117
 to_ulong, 2117
 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference, 2831
 std::try_to_lock_t, 3080
 std::tuple< _Elements >, 3080
 std::tuple< _T1, _T2 >, 3082
 std::tuple_element< 0, pair< _Tp1, _Tp2 > >, 3084
 std::tuple_element< 1, pair< _Tp1, _Tp2 > >, 3084
 std::tuple_element< __i, _Tp >, 3084
 std::tuple_element< __i, tuple< _Types... > >, 3085
 std::tuple_element< _Ind, array< _Tp, _Nm > >, 3085
 std::tuple_size< _Tp >, 3085
 std::tuple_size< array< _Tp, _Nm > >, 3085
 std::tuple_size< pair< _Tp1, _Tp2 > >, 3086
 std::tuple_size< tuple< _Elements... > >, 3087
 std::type_index, 3089
 std::type_info, 3090
 ~type_info, 3091
 before, 3091
 name, 3091
 std::type_order< _Tp, _Up >, 3091
 std::unary_function< _Arg, _Result >, 3096
 argument_type, 3096
 result_type, 3096
 std::unary_negate< _Predicate >, 3097
 argument_type, 3097
 result_type, 3097
 std::underflow_error, 3099
 what, 3099
 std::underlying_type< _Tp >, 3100
 std::uniform_int_distribution< _IntType >, 3100
 max, 3101
 min, 3101
 operator(), 3101
 operator==, 3102
 param, 3102
 reset, 3102
 result_type, 3101
 uniform_int_distribution, 3101
 std::uniform_int_distribution< _IntType >::param_type, 2743
 std::uniform_random_bit_generator, 751
 std::uniform_real_distribution< _RealType >, 3102
 max, 3104
 min, 3104
 operator(), 3104
 operator==, 3105
 param, 3104
 reset, 3105
 result_type, 3103
 uniform_real_distribution, 3104
 std::uniform_real_distribution< _RealType >::param_type, 2744
 std::unique_lock< _Mutex >, 3105
 swap, 3106
 std::unique_ptr< _Tp, _Dp >, 3106
 ~unique_ptr, 3109
 get, 3110
 get_deleter, 3110
 operator bool, 3110
 operator->, 3110
 operator=, 3110, 3111
 operator*, 3110
 release, 3111
 reset, 3111
 swap, 3111
 unique_ptr, 3108, 3109
 std::unique_ptr< _Tp[], _Dp >, 3112
 ~unique_ptr, 3115
 get, 3115

- get_deleter, 3115
- operator bool, 3115
- operator<<, 3117
- operator>, 3116
- operator=, 3116
- operator[], 3116
- operator*, 3115
- release, 3116
- reset, 3117
- swap, 3117
- unique_ptr, 3113–3115
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3120
 - allocator_type, 3124
 - at, 3128–3130
 - begin, 3130, 3131
 - bucket_count, 3131
 - cbegin, 3131
 - cend, 3132
 - clear, 3132
 - const_iterator, 3124
 - const_local_iterator, 3124
 - const_pointer, 3124
 - const_reference, 3125
 - contains, 3132, 3133
 - count, 3133
 - difference_type, 3125
 - emplace, 3133
 - emplace_hint, 3134
 - empty, 3134
 - end, 3135
 - equal_range, 3135, 3136
 - erase, 3136–3138
 - find, 3138, 3139
 - get_allocator, 3139
 - hash_function, 3139
 - hasher, 3125
 - insert, 3139–3142
 - iterator, 3125
 - key_eq, 3143
 - key_equal, 3125
 - key_type, 3125
 - load_factor, 3143
 - local_iterator, 3125
 - mapped_type, 3125
 - max_bucket_count, 3143
 - max_load_factor, 3143
 - max_size, 3144
 - operator=, 3144
 - operator[], 3144, 3145
 - pointer, 3126
 - reference, 3126
 - rehash, 3146
 - reserve, 3146
 - size, 3146
 - size_type, 3126
 - swap, 3146
 - unordered_map, 3126–3128
 - value_type, 3126
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3149
 - allocator_type, 3153
 - begin, 3158
 - bucket_count, 3158
 - cbegin, 3159
 - cend, 3159
 - clear, 3160
 - const_iterator, 3153
 - const_local_iterator, 3153
 - const_pointer, 3153
 - const_reference, 3154
 - contains, 3160
 - count, 3160
 - difference_type, 3154
 - emplace, 3161
 - emplace_hint, 3161
 - empty, 3162
 - end, 3162
 - equal_range, 3163
 - erase, 3163–3165
 - find, 3165, 3166
 - get_allocator, 3166
 - hash_function, 3166
 - hasher, 3154
 - insert, 3166–3169
 - iterator, 3154
 - key_eq, 3170
 - key_equal, 3154
 - key_type, 3154
 - load_factor, 3170
 - local_iterator, 3154
 - mapped_type, 3154
 - max_bucket_count, 3170
 - max_load_factor, 3170
 - max_size, 3171
 - operator=, 3171
 - pointer, 3155
 - reference, 3155
 - rehash, 3171
 - reserve, 3172
 - size, 3172
 - size_type, 3155
 - swap, 3172
 - unordered_multimap, 3155–3157
 - value_type, 3155
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3175
 - allocator_type, 3178

- begin, [3183](#)
- bucket_count, [3184](#)
- cbegin, [3184](#)
- cend, [3184](#)
- clear, [3185](#)
- const_iterator, [3178](#)
- const_local_iterator, [3179](#)
- const_pointer, [3179](#)
- const_reference, [3179](#)
- contains, [3185](#)
- count, [3186](#)
- difference_type, [3179](#)
- emplace, [3186](#)
- emplace_hint, [3186](#)
- empty, [3187](#)
- end, [3187](#), [3188](#)
- equal_range, [3188](#)
- erase, [3189](#), [3190](#)
- find, [3190](#), [3191](#)
- get_allocator, [3191](#)
- hash_function, [3191](#)
- hasher, [3179](#)
- insert, [3191](#)–[3193](#)
- iterator, [3179](#)
- key_eq, [3194](#)
- key_equal, [3179](#)
- key_type, [3180](#)
- load_factor, [3194](#)
- local_iterator, [3180](#)
- max_bucket_count, [3194](#)
- max_load_factor, [3194](#)
- max_size, [3194](#)
- operator=, [3195](#)
- pointer, [3180](#)
- reference, [3180](#)
- rehash, [3195](#)
- reserve, [3196](#)
- size, [3196](#)
- size_type, [3180](#)
- swap, [3196](#)
- unordered_multiset, [3180](#)–[3182](#)
- value_type, [3180](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3199](#)
 - allocator_type, [3202](#)
 - begin, [3207](#)
 - bucket_count, [3208](#)
 - cbegin, [3208](#)
 - cend, [3208](#)
 - clear, [3209](#)
 - const_iterator, [3202](#)
 - const_local_iterator, [3203](#)
 - const_pointer, [3203](#)
 - const_reference, [3203](#)
 - contains, [3209](#)
 - count, [3210](#)
 - difference_type, [3203](#)
 - emplace, [3210](#)
 - emplace_hint, [3210](#)
 - empty, [3211](#)
 - end, [3211](#), [3212](#)
 - equal_range, [3212](#)
 - erase, [3213](#), [3214](#)
 - find, [3214](#), [3215](#)
 - get_allocator, [3215](#)
 - hash_function, [3215](#)
 - hasher, [3203](#)
 - insert, [3215](#)–[3218](#)
 - iterator, [3203](#)
 - key_eq, [3219](#)
 - key_equal, [3203](#)
 - key_type, [3204](#)
 - load_factor, [3219](#)
 - local_iterator, [3204](#)
 - max_bucket_count, [3219](#)
 - max_load_factor, [3219](#)
 - max_size, [3220](#)
 - operator=, [3220](#)
 - pointer, [3204](#)
 - reference, [3204](#)
 - rehash, [3220](#)
 - reserve, [3221](#)
 - size, [3221](#)
 - size_type, [3204](#)
 - swap, [3221](#)
 - unordered_set, [3204](#)–[3206](#)
 - value_type, [3204](#)
- std::uses_allocator< tuple< _Types... >, _Alloc >, [3223](#)
- std::uses_allocator< typename, typename >, [3223](#)
- std::valarray< _Tp >, [3224](#)
 - begin, [3227](#)
 - end, [3227](#)
 - operator!, [3227](#)
 - operator<=, [3229](#)
 - operator>=, [3229](#)
 - operator+, [3228](#)
 - operator+=, [3228](#)
 - operator-, [3228](#)
 - operator=, [3228](#), [3229](#)
 - operator/=: [3229](#)
 - operator%=: [3227](#)
 - operator&=: [3227](#)
 - operator*=, [3228](#)
 - operator~, [3230](#)
 - operator^=: [3230](#)
 - operator|=, [3230](#)
 - valarray, [3227](#)
- std::vector< _Tp, _Alloc >, [3233](#)

- [_M_allocate_and_copy, 3240](#)
- [_M_range_check, 3240](#)
- [~vector, 3240](#)
- [assign, 3240, 3241](#)
- [at, 3241, 3242](#)
- [back, 3242](#)
- [begin, 3242](#)
- [capacity, 3243](#)
- [cbegin, 3243](#)
- [cend, 3243](#)
- [clear, 3243](#)
- [crbegin, 3243](#)
- [crend, 3243](#)
- [data, 3244](#)
- [emplace, 3244](#)
- [empty, 3244](#)
- [end, 3244](#)
- [erase, 3245](#)
- [front, 3245, 3246](#)
- [get_allocator, 3246](#)
- [insert, 3246–3248](#)
- [max_size, 3248](#)
- [operator=, 3248, 3249](#)
- [operator\[\], 3249](#)
- [pop_back, 3250](#)
- [push_back, 3250](#)
- [rbegin, 3250](#)
- [rend, 3251](#)
- [reserve, 3251](#)
- [resize, 3251, 3252](#)
- [shrink_to_fit, 3252](#)
- [size, 3252](#)
- [swap, 3252](#)
- [vector, 3237–3239](#)
- [std::vector< bool, _Alloc >, 3253](#)
 - [_M_allocate_and_copy, 3257](#)
 - [get_allocator, 3257](#)
- [std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3258](#)
 - [_M_buf_locale, 3274](#)
 - [_M_in_beg, 3275](#)
 - [_M_in_cur, 3275](#)
 - [_M_in_end, 3275](#)
 - [_M_out_beg, 3275](#)
 - [_M_out_cur, 3275](#)
 - [_M_out_end, 3275](#)
 - [__streambuf_type, 3260](#)
 - [char_type, 3260](#)
 - [eback, 3262](#)
 - [egptr, 3262](#)
 - [epptr, 3262](#)
 - [gbump, 3262](#)
 - [getloc, 3263](#)
 - [gptr, 3263](#)
 - [imbue, 3263](#)
 - [in_avail, 3264](#)
 - [int_type, 3260](#)
 - [off_type, 3261](#)
 - [overflow, 3264](#)
 - [pbackfail, 3265](#)
 - [pbase, 3265](#)
 - [pbump, 3265](#)
 - [pos_type, 3261](#)
 - [pptr, 3266](#)
 - [pubimbue, 3266](#)
 - [pubseekoff, 3266](#)
 - [pubseekpos, 3267](#)
 - [pubsetbuf, 3267](#)
 - [pubsync, 3267](#)
 - [sbumpc, 3267](#)
 - [seekoff, 3268](#)
 - [seekpos, 3268](#)
 - [setbuf, 3268](#)
 - [setg, 3269](#)
 - [setp, 3269](#)
 - [sgetc, 3269](#)
 - [sgetn, 3270](#)
 - [showmanyc, 3270](#)
 - [snextc, 3271](#)
 - [sputbackc, 3271](#)
 - [sputc, 3271](#)
 - [sputn, 3272](#)
 - [state, 3272](#)
 - [sungetc, 3272](#)
 - [sync, 3272](#)
 - [traits_type, 3261](#)
 - [uflow, 3273](#)
 - [underflow, 3273](#)
 - [wbuffer_convert, 3261](#)
 - [xsgetn, 3273](#)
 - [xsputn, 3274](#)
- [std::weak_ptr< _Tp >, 3275](#)
- [std::weakly_incrementable, 752](#)
- [std::weibull_distribution< _RealType >, 3276](#)
 - [a, 3278](#)
 - [b, 3278](#)
 - [max, 3278](#)
 - [min, 3278](#)
 - [operator\(\), 3278](#)
 - [operator==, 3279](#)
 - [param, 3278](#)
 - [reset, 3279](#)
 - [result_type, 3278](#)
- [std::weibull_distribution< _RealType >::param_type, 2744](#)
- [std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3279](#)
 - [converted, 3281](#)
 - [from_bytes, 3281](#)

- state, 3282
- to_bytes, 3282
- wstring_convert, 3280
- std_abs.h, 3403
- std_function.h, 3404
- std_mutex.h, 3404
- std_thread.h, 3405
- stdatomic.h, 3694
- stdbit.h, 3694
- stdc++.h, 3734
- stdckdint.h, 3695
- stddev
 - std::normal_distribution<_RealType>, 2570
- stdexcept, 3695
- stdexcept_except.h, 3405
- stdexcept_throw.h, 3406
- stdexcept_throwdef.h, 3406
- stdexcept_throwfwd.h, 3406
- stdio_filebuf
 - __gnu_cxx::stdio_filebuf<_CharT, _Traits>, 2947
- stdio_filebuf.h, 3609
- stdio_sync_filebuf.h, 3609
- stdlib.h, 3695
 - abs, 3695
- stdtr1c++.h, 3734
- stl_algo.h, 3406
- stl_algobase.h, 3416
- stl_bvector.h, 3421
- stl_construct.h, 3422
- stl_deque.h, 3422
 - _GLIBCXX_DEQUE_BUF_SIZE, 3423
- stl_function.h, 3423
- stl_heap.h, 3426
- stl_iterator.h, 3427, 3430
- stl_iterator_base_funcs.h, 3431
- stl_iterator_base_types.h, 3432
- stl_list.h, 3432
- stl_map.h, 3433
- stl_multimap.h, 3434
- stl_multiset.h, 3435
- stl_numeric.h, 3436
- stl_pair.h, 3437
- stl_queue.h, 3438
- stl_raw_storage_iter.h, 3439
- stl_relops.h, 3439
- stl_set.h, 3440
- stl_stack.h, 3440
- stl_tempbuf.h, 3441
- stl_tree.h, 3442
- stl_uninitialized.h, 3442
- stl_vector.h, 3443
- stop_token, 3695
- str
 - std::basic_istream<_CharT, _Traits, _Alloc>, 1377
 - std::basic_ostringstream<_CharT, _Traits, _Alloc>, 1476
 - std::basic_stringbuf<_CharT, _Traits, _Alloc>, 1716
 - std::basic_stringstream<_CharT, _Traits, _Alloc>, 1759
 - std::match_results<_Bi_iter, _Alloc>, 2453
 - std::sub_match<_Biter>, 2993
 - stream_iterator.h, 3444
 - streambuf, 3695
 - I/O, 171
 - streambuf.tcc, 3444
 - streambuf_iterator.h, 3445
 - streambuf_type
 - std::istreambuf_iterator<_CharT, _Traits>, 2334
 - std::ostreambuf_iterator<_CharT, _Traits>, 2702
 - streamoff
 - std, 591
 - streampos
 - std, 591
 - streamsize
 - std, 591
 - stride
 - Numeric Arrays, 237
 - string, 3696, 3698, 3699
 - Strings, 279
 - string_conversions.h, 3609
 - string_type
 - std::__cxx11::collate<_CharT>, 1903
 - std::__cxx11::collate_byname<_CharT>, 1908
 - std::messages<_CharT>, 2468
 - std::money_get<_CharT, _InIter>, 2480
 - std::money_put<_CharT, _OutIter>, 2484
 - std::moneypunct<_CharT, _Intl>, 2490
 - std::numpunct<_CharT>, 2685
 - string_view, 3699, 3701
 - string_view.tcc, 3445, 3446
 - stringbuf
 - I/O, 171
 - stringfwd.h, 3446
 - Strings, 279
 - string, 279
 - u16string, 279
 - u32string, 280
 - wstring, 280
 - stringstream
 - I/O, 171
 - strstream, 3293
 - substr
 - __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>, 887
 - __gnu_debug::basic_string<_CharT, _Traits, _Allocator>, 1587

- std::basic_string< _CharT, _Traits, _Alloc >, 1692
- subtract_with_carry_engine
 - std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2996
- subtractive_rng
 - __gnu_cxx::subtractive_rng, 3000
- subview
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1588
 - std::basic_string< _CharT, _Traits, _Alloc >, 1693
- suffix
 - std::match_results< _Bi_iter, _Alloc >, 2453
- sum
 - Numeric Arrays, 237
- sungetc
 - __gnu_cxx::enc_filebuf< _CharT >, 2133
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2961
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2979
 - std::basic_filebuf< _CharT, _Traits >, 1136
 - std::basic_streambuf< _CharT, _Traits >, 1509
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1716
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3272
- swap
 - __gnu_cxx, 414
 - __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 887
 - __gnu_cxx::pair< _T1, _T2 >, 2724
 - __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1588
 - __gnu_parallel::__iteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 927, 928
 - __gnu_pbds::sample_probe_fn, 2864
 - __gnu_pbds::sample_range_hashing, 2865
 - __gnu_pbds::sample_ranged_hash_fn, 2866
 - __gnu_pbds::sample_resize_policy, 2869
 - __gnu_pbds::sample_resize_trigger, 2872
 - __gnu_pbds::sample_size_policy, 2873
 - __gnu_pbds::sample_update_policy, 2877
 - Futures, 121
 - Metaprogramming, 362
 - Numeric Arrays, 237
 - Pointer Abstractions, 353
 - Regular Expressions, 277
 - std, 653–656
 - std::__debug, 665
 - std::basic_regex< _Ch_type, _Rx_traits >, 1494
 - std::basic_string< _CharT, _Traits, _Alloc >, 1694
 - std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT(_Noex)>, 1950
 - std::deque< _Tp, _Alloc >, 2085
 - std::experimental::fundamentals_v1::any, 1086
 - std::forward_list< _Tp, _Alloc >, 2195
 - std::function< _Res(_ArgTypes...)>, 2207
 - std::list< _Tp, _Alloc >, 2386
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2440
 - std::match_results< _Bi_iter, _Alloc >, 2454
 - std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_NOEXCEPT(_Noex)>, 2510, 2511
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2533
 - std::multiset< _Key, _Compare, _Alloc >, 2554
 - std::pair< _T1, _T2 >, 2729
 - std::set< _Key, _Compare, _Alloc >, 2906
 - std::shared_ptr< _Tp >, 2922
 - std::sub_match< _Biter >, 2993
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2116
 - std::unique_lock< _Mutex >, 3106
 - std::unique_ptr< _Tp, _Dp >, 3111
 - std::unique_ptr< _Tp[], _Dp >, 3117
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3196
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3221
 - std::vector< _Tp, _Alloc >, 3252
 - Type-safe container of any type, 298
 - Utilities, 313, 314
- swap_ranges
 - Mutating, 39
- sync
 - __gnu_cxx::enc_filebuf< _CharT >, 2133
 - __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2962
 - __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2979
 - std::basic_filebuf< _CharT, _Traits >, 1136
 - std::basic_fstream< _CharT, _Traits >, 1182
 - std::basic_ifstream< _CharT, _Traits >, 1226
 - std::basic_iostream< _CharT, _Traits >, 1297
 - std::basic_istream< _CharT, _Traits >, 1337
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1378
 - std::basic_streambuf< _CharT, _Traits >, 1509
 - std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1716
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3272
 - sync_with_stdio
 - std::basic_fstream< _CharT, _Traits >, 1182
 - std::basic_ifstream< _CharT, _Traits >, 1227
 - std::basic_ios< _CharT, _Traits >, 1252
 - std::basic_iostream< _CharT, _Traits >, 1297
 - std::basic_istream< _CharT, _Traits >, 1337
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1378

- 1378
- std::basic_ofstream< _CharT, _Traits >, 1411
- std::basic_ostream< _CharT, _Traits >, 1443
- std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1476
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
- std::ios_base, 2303
- syncstream, 3702
- syntax_option_type
 - std::regex_constants, 723
- synth_access_traits
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3078
 - __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3080
- synth_access_traits.hpp, 3583
- system_category
 - Diagnostics, 133
- system_error, 3702, 3703
- t
 - std::binomial_distribution< _IntType >, 1801
- table
 - std::ctype< char >, 1988
 - std::ctype_byname< char >, 2043
- table_size
 - std::ctype< char >, 1996
 - std::ctype_byname< char >, 2047
- tag_and_trait.hpp, 3593
- Tags, 150
 - trivial_iterator_difference_type, 150
- tags.h, 3683
- tan
 - Complex Numbers, 191
 - math.h, 3636
- tanh
 - Complex Numbers, 191
 - math.h, 3637
- target
 - std::function< _Res(_ArgTypes...) >, 2207, 2208
- target_type
 - std::function< _Res(_ArgTypes...) >, 2208
- Technical Specifications, 280
- tell
 - std::basic_fstream< _CharT, _Traits >, 1183
 - std::basic_ifstream< _CharT, _Traits >, 1227
 - std::basic_iostream< _CharT, _Traits >, 1297
 - std::basic_istream< _CharT, _Traits >, 1338
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1378
- std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
- tellp
 - std::basic_fstream< _CharT, _Traits >, 1183
 - std::basic_iostream< _CharT, _Traits >, 1298
 - std::basic_ofstream< _CharT, _Traits >, 1412
 - std::basic_ostream< _CharT, _Traits >, 1443
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1477
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
- temporary_buffer
 - __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3004
- terminate
 - Exceptions, 137
- terminate_handler
 - Exceptions, 135
- test
 - std::bitset< _Nb >, 1820
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2116
- text_encoding, 3703
- text_encoding-data.h, 3446
- tgmath.h, 3703
- thin_heap.hpp, 3587
- this_thread_sleep.h, 3446
- thousands_sep
 - std::moneypunct< _CharT, _Intl >, 2496
 - std::moneypunct_byname< _CharT, _Intl >, 2504
 - std::numpunct< _CharT >, 2688
 - std::numpunct_byname< _CharT >, 2692
- thread, 3704
- Threads, 125
- throw_allocator.h, 3610
- throw_with_nested
 - Exceptions, 137
- tie
 - std::basic_fstream< _CharT, _Traits >, 1183, 1184
 - std::basic_ifstream< _CharT, _Traits >, 1227
 - std::basic_ios< _CharT, _Traits >, 1252
 - std::basic_iostream< _CharT, _Traits >, 1298
 - std::basic_istream< _CharT, _Traits >, 1338
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1379
 - std::basic_ofstream< _CharT, _Traits >, 1412
 - std::basic_ostream< _CharT, _Traits >, 1444
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1477
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1761
 - Utilities, 314
 - Time, 366
 - abs, 376
 - ceil, 376, 377

- clock_cast, 377
- days, 375
- duration_cast, 378
- floor, 378, 379
- high_resolution_clock, 375
- hours, 375
- is_am, 379
- is_pm, 379
- local_time_format, 380
- make12, 380
- make24, 380
- microseconds, 375
- milliseconds, 375
- minutes, 375
- months, 375
- nanoseconds, 376
- operator<, 384
- operator<<, 384
- operator<=, 384
- operator<=>, 385
- operator>, 385
- operator>=, 385
- operator+, 382, 383
- operator-, 383
- operator/, 383, 384
- operator==, 385
- operator%, 382
- operator*, 382
- round, 385, 386
- seconds, 376
- time_point_cast, 386
- weeks, 376
- years, 376
- time
 - std::locale, 2397
- time_get
 - std::time_get< _CharT, _InIter >, 3019
- time_members.h, 3735
- time_point_cast
 - Time, 386
- time_put
 - std::time_put< _CharT, _OutIter >, 3042
- tinyness_before
 - std::__numeric_limits_base, 817
 - std::numeric_limits< _Tp >, 2611
 - std::numeric_limits< bool >, 2615
 - std::numeric_limits< char >, 2619
 - std::numeric_limits< char16_t >, 2623
 - std::numeric_limits< char32_t >, 2627
 - std::numeric_limits< double >, 2631
 - std::numeric_limits< float >, 2635
 - std::numeric_limits< int >, 2639
 - std::numeric_limits< long >, 2643
 - std::numeric_limits< long double >, 2647
 - std::numeric_limits< long long >, 2651
 - std::numeric_limits< short >, 2655
 - std::numeric_limits< signed char >, 2659
 - std::numeric_limits< unsigned char >, 2663
 - std::numeric_limits< unsigned int >, 2667
 - std::numeric_limits< unsigned long >, 2671
 - std::numeric_limits< unsigned long long >, 2675
 - std::numeric_limits< unsigned short >, 2679
 - std::numeric_limits< wchar_t >, 2683
- TLB_size
 - __gnu_parallel::_Settings, 1044
- to_address
 - Pointer Abstractions, 353
- to_array
 - Array creation functions, 288
- to_bytes
 - std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3282
- to_string
 - std::bitset< _Nb >, 1820
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2117
- to_ullong
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2117
- to_ulong
 - std::bitset< _Nb >, 1820
 - std::tr2::dynamic_bitset< _WordT, _Alloc >, 2117
- Todo List, 2
- tolower
 - std, 656
 - std::__ctype_abstract_base< _CharT >, 785, 786
 - std::ctype< _CharT >, 1962, 1963
 - std::ctype< char >, 1988–1990
 - std::ctype< wchar_t >, 2007
 - std::ctype_byname< _CharT >, 2021, 2022
 - std::ctype_byname< char >, 2043, 2044
- top
 - std::priority_queue< _Tp, _Sequence, _Compare >, 2787
 - std::stack< _Tp, _Sequence >, 2943, 2944
- toupper
 - std, 656
 - std::__ctype_abstract_base< _CharT >, 786
 - std::ctype< _CharT >, 1963
 - std::ctype< char >, 1991–1993
 - std::ctype< wchar_t >, 2007, 2008
 - std::ctype_byname< _CharT >, 2022
 - std::ctype_byname< char >, 2044–2046
- TR1 Mathematical Special Functions, 251
 - assoc_laguerre, 253
 - assoc_legendre, 253
 - beta, 253
 - comp_ellint_1, 253
 - comp_ellint_2, 254
 - comp_ellint_3, 254

- cyl_bessel_i, [254](#)
- cyl_bessel_j, [254](#)
- cyl_bessel_k, [254](#)
- cyl_neumann, [254](#)
- ellint_1, [254](#)
- ellint_2, [255](#)
- ellint_3, [255](#)
- expint, [255](#)
- hermite, [255](#)
- laguerre, [255](#)
- legendre, [255](#)
- riemann_zeta, [255](#)
- sph_bessel, [256](#)
- sph_legendre, [256](#)
- sph_neumann, [256](#)
- trace_fnimps.hpp, [3565](#), [3566](#)
- Traits, [152](#)
- traits.hpp, [3551](#), [3552](#)
- traits_type
 - std::basic_ios< _CharT, _Traits >, [1242](#)
 - std::basic_istream< _CharT, _Traits >::sentry, [2883](#)
 - std::basic_streambuf< _CharT, _Traits >, [1498](#)
 - std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
 - std::ostream_iterator< _Tp, _CharT, _Traits >, [2699](#)
 - std::ostreambuf_iterator< _CharT, _Traits >, [2703](#)
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3261](#)
- transform
 - Mutating, [39](#), [40](#)
 - std::__cxx11::collate< _CharT >, [1906](#)
 - std::regex_traits< _Ch_type >, [2846](#)
- transform_exclusive_scan
 - Generalized Numeric operations, [16](#)
- transform_inclusive_scan
 - Generalized Numeric operations, [17](#), [18](#)
- transform_minimal_n
 - __gnu_parallel::Settings, [1044](#)
- transform_primary
 - std::regex_traits< _Ch_type >, [2847](#)
- transform_reduce
 - Generalized Numeric operations, [19](#), [20](#)
- translate
 - std::regex_traits< _Ch_type >, [2847](#)
- translate_nocase
 - std::regex_traits< _Ch_type >, [2847](#)
- traps
 - std::__numeric_limits_base, [817](#)
 - std::numeric_limits< _Tp >, [2611](#)
 - std::numeric_limits< bool >, [2615](#)
 - std::numeric_limits< char >, [2619](#)
 - std::numeric_limits< char16_t >, [2623](#)
 - std::numeric_limits< char32_t >, [2627](#)
 - std::numeric_limits< double >, [2631](#)
 - std::numeric_limits< float >, [2635](#)
 - std::numeric_limits< int >, [2639](#)
 - std::numeric_limits< long >, [2643](#)
 - std::numeric_limits< long double >, [2647](#)
 - std::numeric_limits< long long >, [2651](#)
 - std::numeric_limits< short >, [2655](#)
 - std::numeric_limits< signed char >, [2659](#)
 - std::numeric_limits< unsigned char >, [2663](#)
 - std::numeric_limits< unsigned int >, [2667](#)
 - std::numeric_limits< unsigned long >, [2671](#)
 - std::numeric_limits< unsigned long long >, [2675](#)
 - std::numeric_limits< unsigned short >, [2679](#)
 - std::numeric_limits< wchar_t >, [2683](#)
- tree
 - __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, [3052](#)
- tree_policy.hpp, [3594](#)
- tree_trace_base.hpp, [3589](#)
- trie
 - __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, [3065](#)
- trie_policy.hpp, [3595](#)
- trie_policy_base.hpp, [3590](#)
- trie_string_access_traits_imp.hpp, [3590](#)
- trivial_iterator_difference_type
 - Tags, [150](#)
- true_type
 - Metaprogramming, [361](#)
- truename
 - std::num_punct< _CharT >, [2688](#)
 - std::num_punct_byname< _CharT >, [2692](#)
- trunc
 - std::basic_fstream< _CharT, _Traits >, [1191](#)
 - std::basic_ifstream< _CharT, _Traits >, [1234](#)
 - std::basic_ios< _CharT, _Traits >, [1259](#)
 - std::basic_iostream< _CharT, _Traits >, [1306](#)
 - std::basic_istream< _CharT, _Traits >, [1345](#)
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, [1386](#)
 - std::basic_ofstream< _CharT, _Traits >, [1419](#)
 - std::basic_ostream< _CharT, _Traits >, [1451](#)
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, [1484](#)
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, [1768](#)
 - std::ios_base, [2309](#)
- try_lock
 - Mutexes, [124](#)
- try_to_lock
 - Mutexes, [125](#)
- tuple, [3706](#), [3707](#)
- tuple_cat
 - Utilities, [314](#)
- type
 - __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >, [3052](#)

- [1932](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`, [1933](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >`, [1933](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >`, [1934](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`, [1934](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >`, [1935](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >`, [1935](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >`, [1936](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >`, [1936](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >`, [1937](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >`, [1937](#)
- `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >`, [1938](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >`, [1930](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >`, [1930](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >`, [1931](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >`, [1931](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >`, [1932](#)
- `__gnu_pbds::detail::default_comb_hash_fn`, [2054](#)
- `__gnu_pbds::detail::default_eq_fn< Key >`, [2056](#)
- `__gnu_pbds::detail::default_hash_fn< Key >`, [2057](#)
- `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >`, [2058](#)
- `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >`, [2059](#)
- `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`, [2060](#)
- `__gnu_pbds::detail::default_update_policy`, [2060](#)
- `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >`, [2142](#)
- Metaprogramming, [361](#)
- `std::experimental::fundamentals_v1::any`, [1086](#)
- Type-safe container of any type, [294](#)
- `any_cast`, [295–297](#)
- `swap`, [298](#)
- `type_order_v`
- Variable Templates, [364](#)
- `type_traits`, [3708, 3712](#)
- `type_traits.h`, [3611](#)
- `type_utils.hpp`, [3590](#)
- `typeid`, [3715](#)
- `typeid`, [3740](#)
- `typelist.h`, [3611](#)
- `types.h`, [3683](#)
- `types_traits.hpp`, [3590](#)
- `u16streampos`
- `std`, [591](#)
- `u16string`
- `__gnu_debug`, [422](#)
- Strings, [279](#)
- `u32streampos`
- `std`, [591](#)
- `u32string`
- `__gnu_debug`, [422](#)
- Strings, [280](#)
- `u8path`
- Filesystem, [168](#)
- `uflow`
- `__gnu_cxx::enc_filebuf< _CharT >`, [2134](#)
- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [2962](#)
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [2980](#)
- `std::basic_filebuf< _CharT, _Traits >`, [1137](#)
- `std::basic_streambuf< _CharT, _Traits >`, [1509](#)
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [1717](#)
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3273](#)
- uncaught_exception
- Exceptions, [137](#)
- underflow
- `__gnu_cxx::enc_filebuf< _CharT >`, [2134](#)
- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [2962](#)
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, [2980](#)
- `std::basic_filebuf< _CharT, _Traits >`, [1137](#)
- `std::basic_streambuf< _CharT, _Traits >`, [1509](#)
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [1717](#)
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, [3273](#)

- underlying_type_t
 - Metaprogramming, 362
- unget
 - std::basic_fstream< _CharT, _Traits >, 1184
 - std::basic_ifstream< _CharT, _Traits >, 1228
 - std::basic_iostream< _CharT, _Traits >, 1299
 - std::basic_istream< _CharT, _Traits >, 1338
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1379
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1761
- unicode-data.h, 3447
- unicode.h, 3447
- Uniform Distributions, 246
 - operator<<, 246, 247
 - operator>>, 247
- uniform_int_dist.h, 3449
- uniform_int_distribution
 - std::uniform_int_distribution< _IntType >, 3101
- uniform_real_distribution
 - std::uniform_real_distribution< _RealType >, 3104
- uninitialized_copy
 - Memory, 328
- uninitialized_copy_n
 - Memory, 329
 - SGL, 162
- uninitialized_default_construct
 - Memory, 329
- uninitialized_default_construct_n
 - Memory, 330
- uninitialized_fill
 - Memory, 330
- uninitialized_fill_n
 - Memory, 330
- uninitialized_move
 - Memory, 331
- uninitialized_move_n
 - Memory, 331
- uninitialized_value_construct
 - Memory, 332
- uninitialized_value_construct_n
 - Memory, 332
- unique
 - Mutating, 41
 - std::forward_list< _Tp, _Alloc >, 2195, 2196
 - std::list< _Tp, _Alloc >, 2386
 - std::shared_ptr< _Tp >, 2923
- unique_copy
 - Mutating, 42
- unique_copy.h, 3684
- unique_copy_minimal_n
 - __gnu_parallel::Settings, 1044
- unique_lock.h, 3449
- unique_ptr
 - std::unique_ptr< _Tp, _Dp >, 3108, 3109
 - std::unique_ptr< _Tp[], _Dp >, 3113–3115
- unique_ptr.h, 3449
- unitbuf
 - std, 656
 - std::basic_fstream< _CharT, _Traits >, 1191
 - std::basic_ifstream< _CharT, _Traits >, 1234
 - std::basic_ios< _CharT, _Traits >, 1259
 - std::basic_iostream< _CharT, _Traits >, 1306
 - std::basic_istream< _CharT, _Traits >, 1345
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1386
 - std::basic_ofstream< _CharT, _Traits >, 1419
 - std::basic_ostream< _CharT, _Traits >, 1451
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1484
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1768
 - std::ios_base, 2309
- Unordered Associative, 128
- unordered_map, 3716, 3718
 - std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3126–3128
- unordered_map.h, 3450
- unordered_multimap
 - std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3155–3157
- unordered_multiset
 - std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3180–3182
- unordered_set, 3719, 3721
 - std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3204–3206
- unordered_set.h, 3453
- unsetf
 - std::basic_fstream< _CharT, _Traits >, 1184
 - std::basic_ifstream< _CharT, _Traits >, 1228
 - std::basic_ios< _CharT, _Traits >, 1253
 - std::basic_iostream< _CharT, _Traits >, 1299
 - std::basic_istream< _CharT, _Traits >, 1339
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1380
 - std::basic_ofstream< _CharT, _Traits >, 1413
 - std::basic_ostream< _CharT, _Traits >, 1444
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1477
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1761
 - std::ios_base, 2304
- unshift
 - std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 770
 - std::codecvt< _InternT, _ExternT, _StateT >, 1857
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1857

- 1864, 1865
- std::codecvt< char, char, mbstate_t >, 1872
- std::codecvt< char16_t, char, mbstate_t >, 1879, 1880
- std::codecvt< char32_t, char, mbstate_t >, 1887
- std::codecvt< wchar_t, char, mbstate_t >, 1894, 1895
- std::codecvt_byname< _InternT, _ExternT, _StateT >, 1901
- update_fn_imps.hpp, 3583
- upper_bound
 - Binary Search, 89, 90
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2440, 2441
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2533
 - std::multiset< _Key, _Compare, _Alloc >, 2554, 2555
 - std::set< _Key, _Compare, _Alloc >, 2906, 2907
- uppercase
 - std, 656
 - std::basic_fstream< _CharT, _Traits >, 1191
 - std::basic_ifstream< _CharT, _Traits >, 1234
 - std::basic_ios< _CharT, _Traits >, 1259
 - std::basic_iostream< _CharT, _Traits >, 1306
 - std::basic_istream< _CharT, _Traits >, 1345
 - std::basic_istreamstream< _CharT, _Traits, _Alloc >, 1386
 - std::basic_ofstream< _CharT, _Traits >, 1419
 - std::basic_ostream< _CharT, _Traits >, 1451
 - std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1484
 - std::basic_stringstream< _CharT, _Traits, _Alloc >, 1768
 - std::ios_base, 2309
- use_count
 - std::shared_ptr< _Tp >, 2923
- use_facet
 - Locales, 179
 - std::locale, 2396
 - std::locale::id, 2277
- uses_allocator_args.h, 3455
- Utilities, 302
 - __addressof, 306
 - __invoke, 306
 - addressof, 306
 - declval, 307
 - forward, 307
 - forward_as_tuple, 308
 - forward_like, 309
 - get, 309
 - make_pair, 310
 - make_tuple, 310
 - move, 310
 - move_if_noexcept, 312
 - operator<=>, 312
 - operator==, 312, 313
 - pair, 305
 - piecewise_construct, 314
 - swap, 313, 314
 - tie, 314
 - tuple_cat, 314
 - utility, 3721, 3722
 - utility.h, 3455
- valarray, 3722
 - Numeric Arrays, 225, 226
 - std::valarray< _Tp >, 3227
- valarray_after.h, 3457
- valarray_array.h, 3466
- valarray_array.tcc, 3474
- valarray_before.h, 3475
- valid_prefix
 - __gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, 966
 - __gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >, 969
- value
 - std::error_code, 2150
 - std::error_condition, 2152
 - std::regex_traits< _Ch_type >, 2848
- value_comp
 - std::map< _Key, _Tp, _Compare, _Alloc >, 2441
 - std::multimap< _Key, _Tp, _Compare, _Alloc >, 2534
 - std::multiset< _Key, _Compare, _Alloc >, 2555
 - std::set< _Key, _Compare, _Alloc >, 2907
- value_compare
 - std::set< _Key, _Compare, _Alloc >, 2892
- value_type
 - __gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1775
 - __gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >, 1780
 - __gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1790
 - __gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >, 1793
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >, 2345
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2349
 - std::allocator_traits< _Alloc >, 1065
 - std::allocator_traits< allocator< _Tp > >, 1070
 - std::allocator_traits< allocator< void > >, 1075
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1079

- std::back_insert_iterator< _Container >, [1107](#)
- std::complex< _Tp >, [1912](#)
- std::front_insert_iterator< _Container >, [2201](#)
- std::insert_iterator< _Container >, [2291](#)
- std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, [2331](#)
- std::istreambuf_iterator< _CharT, _Traits >, [2334](#)
- std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, [2340](#)
- std::ostream_iterator< _Tp, _CharT, _Traits >, [2699](#)
- std::ostreambuf_iterator< _CharT, _Traits >, [2703](#)
- std::raw_storage_iterator< _OutputIterator, _Tp >, [2815](#)
- std::set< _Key, _Compare, _Alloc >, [2892](#)
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, [3126](#)
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, [3155](#)
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, [3180](#)
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, [3204](#)
- Variable template for type traits, [298](#)
- Variable Templates, [363](#)
 - is_layout_compatible_v, [364](#)
 - is_pointer_interconvertible_base_of_v, [364](#)
 - type_order_v, [364](#)
- variant, [3722](#)
- vector, [3722–3724](#)
 - std::__debug::vector< _Tp, _Allocator >, [3233](#)
 - std::vector< _Tp, _Alloc >, [3237–3239](#)
- vector.tcc, [3475](#)
- version.h, [3475](#)
- void_pointer
 - __gnu_cxx::__alloc_traits< _Alloc, typename >, [759](#)
 - std::allocator_traits< _Alloc >, [1065](#)
 - std::allocator_traits< allocator< _Tp > >, [1070](#)
 - std::allocator_traits< allocator< void > >, [1075](#)
 - std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, [1079](#)
- void_t
 - Detection idiom, [292](#)
- vstring.h, [3612](#)
- vstring.tcc, [3615](#)
- vstring_fwd.h, [3616](#)
- vstring_util.h, [3616](#)
- wbuffer_convert
 - std::wbuffer_convert< _Codecvt, _Elem, _Tr >, [3261](#)
- wcerr
 - std, [658](#)
- wcin
 - std, [658](#)
- wclog
 - std, [658](#)
- wcout
 - std, [658](#)
- wregex_token_iterator
 - Regular Expressions, [260](#)
- wcsub_match
 - Regular Expressions, [260](#)
- weeks
 - Time, [376](#)
- wfilebuf
 - I/O, [171](#)
- wfstream
 - I/O, [171](#)
- what
 - __gnu_cxx::forced_error, [2174](#)
 - __gnu_cxx::recursive_init_error, [2829](#)
 - __gnu_pbds::container_error, [1939](#)
 - __gnu_pbds::insert_error, [2289](#)
 - __gnu_pbds::join_error, [2342](#)
 - __gnu_pbds::resize_error, [2851](#)
 - std::bad_alloc, [1109](#)
 - std::bad_cast, [1110](#)
 - std::bad_exception, [1111](#)
 - std::bad_function_call, [1112](#)
 - std::bad_typeid, [1114](#)
 - std::bad_weak_ptr, [1115](#)
 - std::domain_error, [2102](#)
 - std::exception, [2155](#)
 - std::experimental::filesystem::v1::filesystem_error, [2168](#)
 - std::experimental::fundamentals_v1::bad_any_cast, [1110](#)
 - std::experimental::fundamentals_v1::bad_optional_access, [1113](#)
 - std::filesystem::filesystem_error, [2170](#)
 - std::future_error, [2216](#)
 - std::invalid_argument, [2294](#)
 - std::ios_base::failure, [2166](#)
 - std::length_error, [2351](#)
 - std::logic_error, [2399](#)
 - std::out_of_range, [2705](#)
 - std::overflow_error, [2714](#)
 - std::range_error, [2801](#)
 - std::regex_error, [2835](#)
 - std::runtime_error, [2863](#)
 - std::system_error, [3002](#)
 - std::underflow_error, [3099](#)
- widen
 - std::__ctype_abstract_base< _CharT >, [787](#)
 - std::basic_fstream< _CharT, _Traits >, [1185](#)
 - std::basic_ifstream< _CharT, _Traits >, [1228](#)
 - std::basic_ios< _CharT, _Traits >, [1253](#)
 - std::basic_iostream< _CharT, _Traits >, [1299](#)
 - std::basic_istream< _CharT, _Traits >, [1339](#)

- `std::basic_istream< _CharT, _Traits, _Alloc >`, 1380
- `std::basic_ofstream< _CharT, _Traits >`, 1413
- `std::basic_ostream< _CharT, _Traits >`, 1444
- `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1478
- `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1762
- `std::ctype< _CharT >`, 1964
- `std::ctype< char >`, 1993–1995
- `std::ctype< wchar_t >`, 2008, 2009
- `std::ctype_byname< _CharT >`, 2023
- `std::ctype_byname< char >`, 2046, 2047
- width
 - `std::basic_fstream< _CharT, _Traits >`, 1185
 - `std::basic_ifstream< _CharT, _Traits >`, 1229
 - `std::basic_ios< _CharT, _Traits >`, 1253, 1254
 - `std::basic_iostream< _CharT, _Traits >`, 1300
 - `std::basic_istream< _CharT, _Traits >`, 1340
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1380, 1381
 - `std::basic_ofstream< _CharT, _Traits >`, 1413
 - `std::basic_ostream< _CharT, _Traits >`, 1445
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1478
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1762
 - `std::ios_base`, 2304
- wfstream
 - I/O, 171
- wios
 - I/O, 171
- wiostream
 - I/O, 171
- wistream
 - I/O, 171
- wstringstream
 - I/O, 171
- wofstream
 - I/O, 172
- workstealing.h, 3684
- wostream
 - I/O, 172
- wostringstream
 - I/O, 172
- wregex
 - Regular Expressions, 260
- write
 - `std::basic_fstream< _CharT, _Traits >`, 1186
 - `std::basic_iostream< _CharT, _Traits >`, 1300
 - `std::basic_ofstream< _CharT, _Traits >`, 1414
 - `std::basic_ostream< _CharT, _Traits >`, 1445
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1479
- `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1763
- ws
 - std, 657
- wsregex_token_iterator
 - Regular Expressions, 260
- wssub_match
 - Regular Expressions, 261
- wstreambuf
 - I/O, 172
- wstreampos
 - std, 592
- wstring
 - Strings, 280
- wstring_convert
 - `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`, 3280
- wstringbuf
 - I/O, 172
- wstringstream
 - I/O, 172
- xalloc
 - `std::basic_fstream< _CharT, _Traits >`, 1186
 - `std::basic_ifstream< _CharT, _Traits >`, 1229
 - `std::basic_ios< _CharT, _Traits >`, 1254
 - `std::basic_iostream< _CharT, _Traits >`, 1301
 - `std::basic_istream< _CharT, _Traits >`, 1340
 - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1381
 - `std::basic_ofstream< _CharT, _Traits >`, 1414
 - `std::basic_ostream< _CharT, _Traits >`, 1446
 - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1479
 - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 1763
 - `std::ios_base`, 2304
- xsgn
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2134
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2963
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2980
 - `std::basic_filebuf< _CharT, _Traits >`, 1137
 - `std::basic_streambuf< _CharT, _Traits >`, 1510
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1717
 - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3273
- xspn
 - `__gnu_cxx::enc_filebuf< _CharT >`, 2135
 - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 2963
 - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 2981
 - `std::basic_filebuf< _CharT, _Traits >`, 1138
 - `std::basic_streambuf< _CharT, _Traits >`, 1510
 - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 1718

`std::wbuffer_convert<_Codecvt, _Elem, _Tr >`, [3274](#)

years

Time, [376](#)

yield

`std::this_thread`, [734](#)